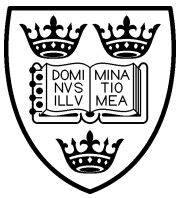
Teaching mobile robots to use spatial words



Simon Dobnik
The Queen's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2009

In loving memory of my father Ivan

Acknowledgements

This work would not have been possible without the support of friends and colleagues whom I would like to thank here.

Firstly, I would like to thank my supervisor, Professor Stephen Pulman, for his support throughout, for his comments and advice at important stages in the development of this thesis and for arranging regular meetings to read my work. Thanks also goes to Paul Newman, Dave Cole, Ingmar Posner, Mark Cummins and the rest of the group for introducing me to mobile robotics and allowing me to work with their equipment and in their lab.

The Centre for Linguistics provided a stimulating and pleasant research environment. I would like to thank David Cram, Sandra Paoli, Professor John Coleman and Professor Mary Dalrymple for allowing me to work there and be part of the academic community. A special thanks goes to Professor Aditi Lahiri for her incredible kindness, guidance and encouragement during the most difficult times.

I would also like to thank my colleagues and friends Jana Sukkarieh, Maria Liakata, David Wright, Peet Morris, Rada Mihalcea and Miltiadis Kokkonidis with whom I have spent a significant part of my DPhil life by sharing an office. They were the first people with whom I discussed my work and ideas and they also supported me morally when things did not go right. I am also grateful to Lars Larm, Keith Owen, Wolfgang de Melo, Jakob Leimgruber and Lindsay Weichel for our discussions and friendship.

The first two years of this study were funded in part by the studentships from the Oxford University Committee for Comparative Philology and General Linguistics, The Queen's College and an ORS Award. I am also grateful to the Ad Futura Foundation

and Gorenje d.d. for their grants. A special thanks goes to The Queen's College for their support in financial hardship, in particular to Jane Langdale and Joyce Millar for their kind assistance.

My appreciation also goes to my friends and colleagues at the Oxford University Computing Services where I spent a significant time when not working on my thesis. These are Catherine Craddock, Peter Higginbotham, Stuart Robeson, Annie Martin, John Howard, Catrin Radcliffe, Andrew Hutchinson, Jeffrey Snyder, Jonathan Sargeant, Tim Fernando, Samira Kazan, Ranjit Vijayan and many others.

I am grateful to my housemates who made my life happier and richer: Stephen Patrick, Elizabeth Harre, Penny Grant and Cheryl Ridout. A special thanks goes to Susanne Sklar, a dear friend and truly an exceptional person.

I could also count on the unfailing support of my family, in particular my mother Milena, my grandmother Roza, Matej and Viljem. The biggest thanks goes to Lidija. For many years she has believed in me, supported me and loved me.

Abstract

The meaning of spatial words can only be evaluated by establishing a reference to the properties of the environment in which the word is used. For example, in order to evaluate what is *to the left of* something or how *fast* is *fast* in a given context, we need to evaluate properties such as the position of objects in the scene, their typical function and behaviour, the size of the scene and the perspective from which the scene is viewed.

Rather than encoding the semantic rules that define spatial expressions by hand, we developed a system where such rules are learned from descriptions produced by human commentators and information that a mobile robot has about itself and its environment. We concentrate on two scenarios and words that are used in them. In the first scenario, the robot is moving in an enclosed space and the descriptions refer to its motion ("You're going forward slowly" and "Now you're turning right"). In the second scenario, the robot is static in an enclosed space which contains real-size objects such as desks, chairs and walls. Here we are primarily interested in prepositional phrases that describe relationships between objects ("The chair is to the left of you" and "The table is further away than the chair"). The perspective can be varied by changing the location of the robot. Following the learning stage, which is performed offline, the system is able to use this domain specific knowledge to generate new descriptions in new environments or to "understand" these expressions by providing feedback to the user, either linguistically or by performing motion actions.

If a robot can be taught to "understand" and use such expressions in a manner that would seem natural to a human observer, then we can be reasonably sure that

we have captured at least something important about their semantics. Two kinds of evaluation were performed. First, the performance of machine learning classifiers was evaluated on independent test sets using 10-fold cross-validation. A comparison of classifier performance (in regard to their accuracy, the Kappa coefficient (κ), ROC and Precision-Recall graphs) is made between (a) the machine learning algorithms used to build them, (b) conditions under which the learning datasets were created and (c) the method by which data was structured into examples or instances for learning. Second, with some additional knowledge required to build a simple dialogue interface, the classifiers were tested live against human evaluators in a new environment. The results show that the system is able to learn semantics of spatial expressions from low level robotic data. For example, a group of human evaluators judged that the live system generated a correct description of motion in 93.47% of cases (the figure is averaged over four categories) and that it generated the correct description of object relation in 59.28% of cases.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural languages contain a number of words whose meaning is underspecified. In order to fully determine the meaning of expressions such as *slowly*, *turning* and *left* in the examples below, additional semantic representations must be constructed.

(1) a. Marge is *slowly turning left*.

b. The green Mini is *slowly turning left*.

(2) a. The table is *to the left of* the chair.

b. The library is *to the left of* the theatre.

The reference of these expressions is resolved by relying on human perception (Miller and Johnson-Laird, 1976). One needs to examine the scene that the descriptions refer to visually to derive their truth conditions – a task known as *grounding* (Roy, 2002). If the hearer cannot examine the scene, the speaker must provide extra natural language descriptions which provide enough information for the hearer to construct a mental representation of the scene. Such expressions thus have some properties of deictic referents (Levinson, 1983, page 79ff.) and (Maillat, 2003, page 39ff.).

For example, in order to determine the reference of "slowly" in (1) one needs to evaluate the size of the objects (Marge, the green Mini) and the size of the environment in which the moment takes place (the room, the street). If we measure the speed of both vehicles referred to by the descriptions, we may conclude that the speed of Marge, a

mobile robot typically used in indoor environments, is considerably lower ($\approx 0.3m/s$) than the speed of the green Mini ($\approx 5.56m/s = 20km/h$), although the same description "slowly" is used in both cases. The region denoted by the relation "to the left of" in (2) is underspecified in the same way: it is also constrained by the size of the objects (the table, the chair, the library and the theatre) and the size of the scene (the room, the street). Furthermore, it may also be constrained by the location and size of the other objects in the scene, their shape and their typical function (Herskovits, 1986). The description, also known as a *directional* (Maillat, 2003), must also be interpreted in relation to a particular *frame of reference*. Are the speaker/hearer sitting in the chair when the speaker makes the utterance or are they observing the scene from some other location in the room? Similarly, in (1), are they inside the green Mini or observing the scene from the pavement?

The descriptions in (1) refer to *dynamic* or *motion* scenes whereas descriptions in (2) refer to *static* scenes. However, since the reference of both is fixed by some properties of the physical space, we group them under the same broader category of *spatial expressions*. They differ in one important aspect: the semantic representations of dynamic scenes require yet another conceptual component: a one-dimensional path along which the entities move (Miller and Johnson-Laird, 1976, page 405ff). Thus *left* in (1) denotes a relation between two locations of a single entity, one at the time t and the other at the time $t + 1$, whereas *left* in (2) denotes a relation between the locations of two entities both at the time t.

Psycholinguistic and anthropological research agree (Miller and Johnson-Laird (1976, page 375), Talmy (2000, page 177ff.) and Levinson (2003, page xvii)) that spatial cognition represents the core of human cognition. It not only allows humans to reason and refer to space in which they live but it has been extended to other cognitive areas as evidenced by many spatial metaphors that we use, for example in diagrams.[1] It

[1]The point where opinions diverge is whether spatial concepts are universal (Miller and Johnson-Laird, 1976; Talmy, 2000) or whether they are culturally specific (Levinson, 2003).

is therefore no surprise that spatial cognition is one of the core areas that has been researched by artificial intelligence and computational linguistics when developing robotic systems or embodied agents which interact with humans and their environment in a natural, human way, for example, Shakey: Nilsson (1984), Shrdlu: Winograd (1976), Vitra: Stopp et al. (1994), Godot: Theobalt et al. (2002) and Bos et al. (2003) and CoSy: Zender et al. (2008). Because such agents connect language with their perception of environment and hence must combine different representations and modes of reasoning they are also known as *multi-modal agents*. They provide a great practical utility: for example they are used as assistive aids focused on individual tasks such as navigation or they can be explorers of dangerous environments.

Of course, such systems do not attempt to re-create artificially the *actual* cognitive processes that take part in the human brain when perceiving and evaluating the environment, the mechanisms that constitute human competence, but they focus on re-creating human behaviour resulting from such competence, thus human performance. In doing so, they rely on methods and instruments that were developed by physical sciences to describe the world. Descriptions of space created using these methods are quite different from human descriptions. All measures introduced by physical sciences are *continuous* in nature. For example, the location of two points can be described by introducing a two-dimensional coordinate system with a fixed point of origin and by specifying their coordinates from the scale of real numbers. On the other hand, natural language expressions use discrete reference to refer to events and objects. They partition space into regions such as *near*, *back* and *left*, and degrees of motion to *slowly*, *moderately* and *fast*. While non-linguistic reference can be made with a high degree of accuracy, the reference to spatial descriptions is ambiguous and vague (Mukerjee, 1998). Thus, the problem is yet another case of a well known problem in computational linguistics: how to pair ambiguous and compact (and hence expressive) descriptions of natural language with unambiguous but consequently syntagmatically more complex

(and hence inexpressive) descriptions created by different formalisms of knowledge representation (for a study see Sukkarieh, 2000). Whereas in such cases the challenge is to find a mapping between two symbolic systems, in spatial reasoning the mapping must be done between a non-symbolic and a symbolic system.

The mapping between the domains is commonly performed by first identifying some parameters of the physical world on the basis of psychological evidence and then integrating them into customised functions (Regier and Carlson, 2001). In contrast, our approach starts with a simple representation of space that is available to us through the sensory data of a mobile robot and through structured non-symbolic maps produced by a technique known as simultaneous localisation and map building or SLAM (Smith and Cheeseman, 1986). The data in this representation is not specifically tailored to the task. We rely on machine learning algorithms to induce associations between such simple representations and linguistic expressions. The models that the classifiers build automatically are considerably different from the models designed by humans. They depend on the structure of the input data and the chosen classification method. For the learning we use the Weka toolkit (Witten and Frank, 2005) which provides us with a range of classifier implementations and a common framework to represent the data and evaluate the results. The objective of the learning is to show that models induced by machine learning are able to replicate human linguistic performance in new environments. This way we can be sure that we captured some important facts about their semantics. To demonstrate a human-like performance, the classifiers must be integrated in a system which is capable of some basic interaction with humans. Two kinds of evaluation were performed: the evaluation of the classifiers on similar datasets from which they were built and the evaluation of the system containing these classifiers by human observers in a real environment.

Our work is practical in nature. Learning and generation of natural language referential descriptions is integrated with a system that navigates and drives a mobile robot.

Working with mobile robots has a number of benefits which can also be viewed as challenges. Firstly, it gives us the opportunity to work in a natural environment with real size objects such as chairs, desks and cupboards. Virtual environments may be suited for theoretical studies of spatial cognition where the parameters must be carefully controlled (Maillat, 2003). However, any practically useful system should be able to deal with real environments which also present the ultimate challenge for its performance. Secondly, the variety of sensory data available to us gives us a very accurate representation of the state of the robot and its surrounding environment which we would otherwise not have. Experiments on real environments may thus reveal important insights and challenges for spatial cognition that have been overlooked before. Thirdly, any system that drives a mobile robot must be *multi-modal*. It must contain a component that collects and processes information from its sensors and a component which provide the robot's response through its actuators, for example, by moving its wheels. Thus, an interesting research question is how such modalities can be integrated, what information can be exchanged and what are the optimal mechanisms of such exchange.

This thesis is organised as follows. In Chapter 2 we discuss the concepts, approaches and tools related to our work. In Chapter 3 we describe how datasets of linguistic and non-linguistic observations were collected and used for machine learning, what algorithms were employed and how successful the learning was. Chapter 4 describes how classifiers were integrated into two systems that provide basic interaction with humans. Chapter 5 discusses the evaluation of these systems by humans in real environments. The last chapter gives some concluding remarks.

Chapter 2

Research background

2.1 Introduction

This chapter contains an overview of topics related to our work. We start with a discussion of semantics and pragmatics of descriptions of object relations which receive considerable attention in the literature. We continue with approaches that attempt to link language with robotic perception of the world. Finally, we describe the tools that we have at hand: the robotic system, the data available to the robot through its sensors, and how generalisations about such data can be made using machine learning.

2.2 Descriptions of object relations

In (1) the descriptions "to the left of" and "near" refer to a relation between two entities "the table" and "the chair". The location of the second entity "the chair" is assumed to be known and is used to describe the location of the first entity "the table". For this reason, the literature refers to these entities as *referent* and *relatum* (Miller and Johnson-Laird, 1976), *figure* and *ground* (Talmy, 1983) or the *located object* (LO) and the reference object (REFO) (Herskovits, 1986; Gapp, 1994a).

(1) a. The table is *to the left of* the chair.

b. The table is *near* the chair.

There are two subgroups of descriptions of *object relations* which have different semantic properties. The description "to the left of" is known as a *projective relation* or a *directional* because it also includes the notion of a *reference frame* as a part of its semantics. The reference frame is not required by *topological relations* such as "near" which refer to a proximal region spanning around the reference object. The presence of the reference frame in directionals is evidenced by the fact that these relations are not symmetric: from (1a) we cannot infer (2a), whereas from (1b) we can infer (2b).

(2) a. The chair is *to the left of* the table.

b. The chair is *near* the table.

The reference frame of directionals can be fixed in three different ways. Consider the following examples all of which describe the same spatial scene shown in Figure 2.1.

(3) a. The table is *behind* the chair.

b. The table is *to the left* of the chair.

c. The table is *north of* the chair.

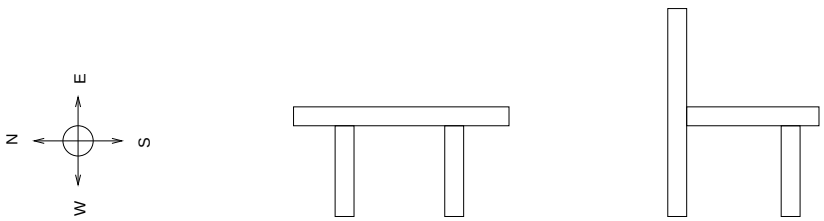


Figure 2.1: Where is the table?

In this scene (3a) is interpreted according to the *intrinsic reference frame*. The location of the table is specified in respect to the reference frame defined by the reference object "the chair". This is possible because the chair has an identifiable front, its sitting area, and a back. However, this cannot be taken as a general rule. Maillat (2003, page 50ff.)

points out that the ability of objects to ground a reference frame cannot be attributed to their innate characteristics, but is assigned to them through a complex cognitive process in which the relevant features of the object are identified. When a description is interpreted according to the intrinsic frame, the locations of the speaker and hearer do not affect the truth conditions of the sentence. This means they can freely move or rotate around the room without affecting its interpretation.

(3b) is true in respect to the scene in Figure 2.1 if the sentence is interpreted according to the *relative reference frame*. This reference frame is fixed by a third point in the scene called a *viewpoint* (VPT) (Maillat, 2003, page 54) different from the located object and the reference object. The location of the viewpoint is often taken as the location of the speaker or the hearer, but this is not necessarily the case. The sentence in (3b) can be followed by any of the following phrases.

(4) a. ...from my/your point of view.

b. ...from the robot's point of view.

c. ...from there.

Most frequently the viewpoint is linguistically unspecified. Instead, the communication participants rely on pragmatics to resolve the location of this third variable which can potentially be any point in the environment. When a relative reference frame is intended, changing the location of the viewpoint around the *array* containing the reference and the located object will affect the truth conditions of the sentence, whereas with the intrinsic reference frame they should remain unchanged. This criterion can be therefore used as a test to determine which reference frame is intended in the interpretation of a description (see Levinson, 1996a, Maillat, 2003, page 47).

Furthermore, since the intended reference frame is also not specified in a sentence, a description such as (3b) can be interpreted either in respect to the intrinsic or the relative reference frame and depending on this a different spatial region will be referred

to. Figure 2.2 shows how descriptions "left", "right", "front" and "back" idealistically partition space around the reference object depending on the choice of the reference frame. It can be seen that the regions denoted by "left" and "right" are reversed, a phenomenon known as a *lateral shift* (Maillat, 2003, page 60, attributing the description of phenomena to Vandeloise, 1986 and Herskovits, 1986). However, the lateral shift is not present in all languages that have the above projective descriptions. For example, in Hausa and Tamil (cited in Maillat, 2003, page 60) the same projective is used to refer to the region regardless which of the two reference frames is used.[1]

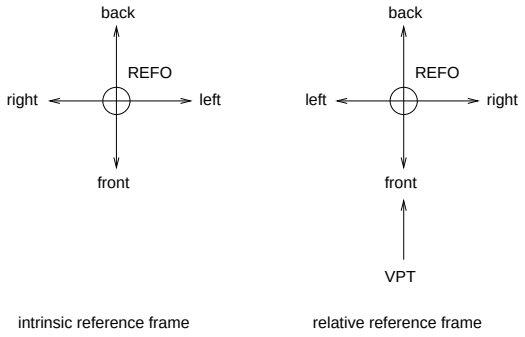


Figure 2.2: "Ideal" regions denoted by projective descriptions depending on the choice of a reference frame

Finally, (3c) must be interpreted according to the *absolute reference frame* when referring to the scene in Figure 2.1. The absolute reference frame is fixed to a secondary reference 'object' which in this case is the North Pole. It follows that the truth conditions of the sentence are unaffected if the reference object or the viewpoint are rotated. The truth conditions only change if we rotate the secondary reference object around the *extended array* containing the reference object, the located object and the viewpoint (Maillat, 2003, page 68).

The projective descriptions "north", "south", "east" and "west" unambiguously imply this reference frame. On the other hand, the pair "above" and "below" is potentially ambiguous between all three reference frames (Maillat, 2003, page 65ff.). How-

[1] Interestingly, the order of the regions denoted by projectives around the centre of the REFO is different between the Indo-European languages, Hausa and Tamil.

ever, the interpretation in respect to the absolute reference frame is by far the most common since in most cases the relative and the intrinsic reference frames coincide with it. This is because the absolute reference is fixed by the Earth's core or the gravity which in turn fixes the vertical location of other objects and viewpoints.

Descriptions such as (3c) are fairly unusual for speakers of English, but there are nonetheless some specialised situations where descriptions using this reference frame are more frequently used. Maillat (2003, page 65) points out that directions in US cities are described relative to an absolute grid, and also that Lake Michigan may be used as an absolute landmark in the Chicago area. On the other hand, Levinson (1996b) describes two languages where the absolute framework is the predominant one. In Guugu–Yimithirr, an Australian aboriginal language spoken in North Queensland, the absolute reference frame is fixed to the North. In Tzeltal, a Mayan language spoken in Mexico, the absolute frame of reference is fixed by an uphill-downhill inclined plane corresponding to a fall of terrain from the South to the North. Thus, even if the two objects are on a flat surface, the one closer to the north will be described as "downhill".

Examples in (3) describe static scenes. However, directionals such as "left", "right", "forward", "backward", "up" and "down" can also be used in dynamic scenes that involve movement as in:

(5) Marge is moving *forward* and to the *left*.

Such descriptions add another conceptual component: a temporal dimension. Whereas in static scenes "left" denotes a relation between locations of two different entities at time t and the location of one entity (the REFO) is used to describe the location of the other (the LO), in dynamic scenes it denotes a relation between two locations of a single entity at times t and $t + 1$. Maillat (2003, footnote 23, page 51) points out that in dynamic scenes motion, which is directed, adds an intrinsic orientation to objects. For this reason such expressions can only be interpreted in respect to the intrinsic reference frame. The interpretation with the relative reference frame is blocked.

The literature presents various models that attempt to account for the preceding facts related to spatial reference frames. Here we briefly outline the proposal in (Maillat, 2003) whose distinguishing characteristic is that it attempts to identify only the minimal conceptual elements that such a model requires.[2] According to this proposal, the reference frames contain the following primitives: (i) a set of three orthogonally arranged and intersected axes, (ii) a point of origin, and (iii) an orientation vector.

The set of axes represent the three dimensions along which directionals can very: the frontal, the lateral and the vertical. The axes intersect in the point called the origin (ORIG). According to Maillat (2003) this is always fixed by the reference object, regardless of the reference frame. This is in contrast to approaches such as (Levinson, 1996a; Herskovits, 1986; Talmy, 1983; Vandeloise, 1986) who assume that in the relative framework there are in fact two origins: one projecting from the reference object and the other projected from the viewpoint. The former is subordinate to the latter since the speaker must "translate" the set of axes with a centre on the viewpoint onto the set of axes centred on the reference object. Maillat (2003) argues that there is no need for such a complex model. Instead, he proposes that in the relative reference frame the viewpoint only assigns the orientation of the axes projected from the reference object (ORIENT) rather than an entire new set of axes.

The model that defines the framework only presupposes two variables: ORIG and ORIENT. Table 2.1, adapted from (Maillat, 2003, Table 5, page 112), shows their values with various reference frames. Since ORIG always corresponds to REFO, it is always linguistically salient but also has no effect on the choice of the reference frame. Instead, this is dependent on the value of the ORIENT parameter. In English and other Indo-European languages, the ORIENT parameter is only partially linguistically realised. It can be recovered from our lexical knowledge of the REFO or it is a part of lexical specifications of some words such as "south" and "east". However, in the presence

[2]For the relation of this proposal to other proposals, in particular to (Levinson, 2003), the reader is referred to (Maillat, 2003, Chapter 3: Primitives and Unification, page 71ff.).

of the VPT it must be pragmatically resolved from the context. Since the VPT is not linguistically salient it contributes to the ambiguity of sentences as (3a) and (3b).

	ORIG	ORIENT
Absolute	REFO	landmark
Intrinsic	REFO	REFO
Relative	REFO	VPT

Table 2.1: The values of the ORIG and ORIENT parameters for different reference frames

Resolution of the reference frame for a particular description with a directional contributes only half of its meaning. Directionals also have a semantic value: they denote a two- or three-dimensional field projected in space that is constrained by the dimensions of the reference frame. In the literature, such fields are known as *spatial templates* or *spatial maps* (Logan and Sadler, 1996, page 496). The fields are not uniform but are graded. The grades show how acceptable it is to relate each of the points to the REFO by the specified relation. Logan and Sadler (1996) generalise them to three discrete grades of acceptability: areas containing good examples, areas containing less good but nonetheless acceptable examples and areas containing unacceptable examples. Figure 2.3 shows a two-dimensional spatial template for the description "above".

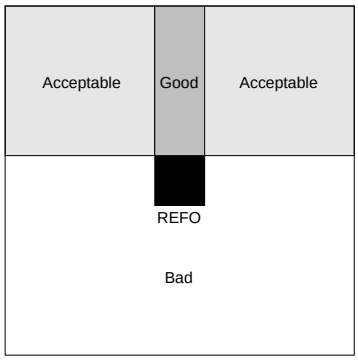


Figure 2.3: Spatial template for "above" (Logan and Sadler, 1996, page 497)

In most cases, spatial templates are derived experimentally. For example, Logan and Sadler conduct experiments where participants are presented with images con-

sisting of a frame representing a spatial scene. The frame is subdivided into a grid consisting of 7 columns by 7 rows which is not shown to the participants. In each case, the mid-cell of the grid (4,4) is occupied by an "O" representing the REFO. The participants have to rate the appropriateness of a given spatial description relating the "O" (REFO) and an "X" (LO) which is randomly placed in one of the remaining 48 cells of the grid using a 10 point scale. Subsequently, the mean rating is calculated for each of the 48 cells and the results are generalised as in Figure 2.3.[3] Logan and Sadler also show that the highest ratings are consistent with results obtained in the production task in which the participants had to mark the location of an X given a spatial description such as "Draw an X above the box".

This experimental design is quite popular in literature (see for example Kelleher and Costello, 2005). However, it has one shortcoming, namely it does not use real three-dimensional space but its projection onto a two-dimensional plane. This is overcome in the work of Maillat (2003) who uses computer-generated three-dimensional images of a virtual environment. The environment consists of a chequered surface with two marbles also referred to as snooker balls. The marble that is used as the REFO is generated at the centre of each three-dimensional scene. The other marble, representing the LO, is generated so that is appears equidistant to the REFO between the scenes, but in each scene of a sequence it is rotated by some increment of degrees around or above the REFO. The angular value can be converted to the x and y dimensions of a plane using the sine and cosine functions (see Maillat, 2003, page 156).

Logan and Sadler do not see spatial templates just as semantic models of spatial relations but as psychological concepts that humans employ in their apprehension. They argue, for example, that in order to check whether a particular spatial relation applies to describe two objects, humans map its spatial template within the reference frame projected from the REFO and ascertain the goodness of fit of the LO. If a particular

[3]Logan and Sadler also give three-dimensional representations of spatial templates where the mean ratings are represented as values on the z axis.

LO falls within the field of two spatial templates, then the template with a better fit is chosen (Logan and Sadler, 1996, page 497).

There is a distinct asymmetry between good/acceptable and bad regions in Figure 2.3. This can be explained by the fact that spatial templates are fixed on the axes of a reference frame which constrains the frontal and the lateral dimensions. Because they cover an entire dimension, Maillat (2003) refers to them as maximal templates. Within each dimension spatial templates show a graded transition from good to acceptable regions. This means that two templates such as "to the right of" and "in front of" inevitably overlap in one quadrant of the reference frame. Both facts are confirmed by Maillat's experimental work. He shows that the templates provide the best fit around the corresponding axes and that one template smoothly transits into another as the rotation progresses (Maillat, 2003, Figure 14, page 149).

Up to now the spatial templates were only defined in geometrical terms and for this reason only abstract objects were used in the experiments. However, the function of objects and the way they interact with each other and the way we interact with them may also be important in determining the appropriateness of an expression. Coventry (2003) discusses experiments[4] where participants had to rate descriptions containing objects with a function, for example, an umbrella held by a man at angles of 0, 45 and 90 degrees anti-clockwise from the vertical plane. It was shown that although the overall acceptability ratings of "above" decreased with increased rotation, the highest overall ratings were given for a set of scenes where the umbrella was providing protection from the rain. These were followed by the ratings of scenes with no rain and finally by the ratings of scenes with the rain but where the umbrella was not providing protection (Coventry, 2003, Figure 13.2, page 260). Furthermore, Coventry also refers to experiments that show that function and geometry affect spatial relations to a different degree: a certain relation may be influenced more by one manipulation than the other.

[4]These are described in detail in (Coventry et al., 2001).

The ways in which geometric and functional information may be integrated in a model is still a matter for future research.

There are also proposals of models that generate spatial templates (Olivier and Tsujii, 1994; Gapp, 1994a,b, 1995; Regier, 1996).[5] A model is a set of functions that predict acceptability ratings for each point of space surrounding the REFO. This is why in such cases spatial templates may be referred to as *potential fields* (Olivier and Tsujii, 1994, page 306). The parameters of models vary between different approaches and include some measure of distance and angle of orientation between the objects. For example, Regier (1996, page 82ff.) considers *proximal orientation* and *center-of-mass orientation* between the LO and the REFO to be important. The proximal orientation is the angle defined by the lines between the closest points between the objects and one of the axes of the reference frame, whereas the center-of-mass orientation is the angle defined by the lines between the centre points of objects and one of the axes, as shown in Figure 2.4. The angles measure deviations from the axes of the reference frame that mark the perfect acceptability of a particular expression. This means that for "above" or "behind" the relevant axis is the positive vertical axis, whereas for "right" this is the positive horizontal one.

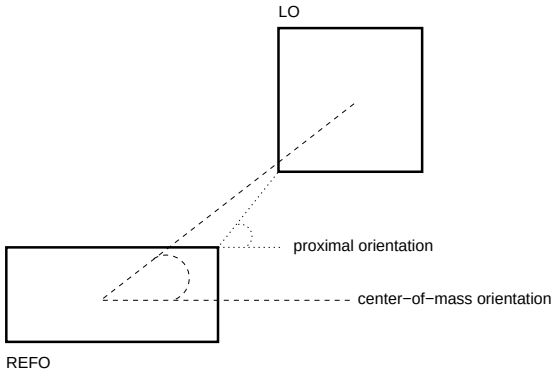


Figure 2.4: Two orientation components

[5]See also the discussion of the "overlap theory" and the "quadrant theory" in (Maillat, 2003, page 133ff.).

Regier and Carlson (2001) perform empirical validation of the most common models found in the literature. They also propose and validate a new model which they call the *attentional vector sum model* (AVS) which combines the intuitions behind the previous models. The model is built of two components both of which, as the authors claim, have grounding in biological systems. Following the work of Logan (1994, 1995) they propose that the apprehension of spatial relations is an active process which requires *attention*. Therefore, the spatial templates are not predefined but instead are constructed during the perception of a scene. In the AVS the attention is modelled as an attentional field, a graded beam fixed at the point of the REFO which is closest to the LO. The attentional field is distributed over the REFO and thus some parts, those closest to the beam's origin, receive more attention than the others. The second component of the model is the notion of a *vector sum*. For each point of the REFO the model defines a vector pointing toward the centre of the LO. Each vector is weighted by the attention of its point of origin. In the case of "above" the sum of the vectors is then compared with the positive vertical axis (Regier and Carlson, 2001, Figure 5, page 278). The attentional vector sum thus combines the effects of proximal and center-of-mass orientations. In addition, the AVS model also incorporates two other features proposed in other models, namely the effect of the grazing-line[6] and the distance between the objects. Its experimental validation shows that it performs better than its predecessors.

For topological descriptions a different model is required. Kelleher et al. (2006) present a model of proximity for "near". The model evaluates the proximal distance between the REFO and each point of the grid weighted by the REFO's visual and discourse salience. The proximity of the REFO to a point is expressed relative to the proximity of other competing REFOs to that point. In this way, the best REFO to describe the proximity of a point can be selected.

[6]This is the line that for "above" runs across the top of a rectangular REFO, thus the side that is closest to the LO. High acceptability ratings are given to points just above this line, and considerably lower acceptability ratings are given to points just below this line.

2.3 Natural language and situated robots

Natural language is an expressive and efficient way by which humans share information. A robot that can interact with humans in a natural language is thus of great practical utility, especially if human users are not expert roboticists. The literature describes many such systems, for example (Winograd, 1976, Stopp et al., 1994, Theobalt et al., 2002 and Zender et al., 2008) to name just a few. The key feature of a conversational robot is that its language must be situated. The robot must be able to refer to and resolve references to the environment in which it is located with its human conversational partner. Since every real environment is uncertain or changing, machine learning plays an important role in building both linguistic and non-linguistic representations.

Robots perceive the environment differently than humans do. The conceptual representations resulting from the robotic perception are intended primarily for tasks such as localisation and navigation and not for generating and understanding linguistic descriptions. Both non-linguistic tasks may be accomplished by considering clouds of points represented in a two- or three-dimensional space obtained by laser scans. Linguistic descriptions on the other hand refer to higher-level conceptual elements such as directions, regions or even confined units of space such as rooms. Of course, robotic perception can be enriched by adding different types of sensors. However, these produce different conceptual representations of the environment which must be fused somehow. For example, a video camera provides colour images which only cover two, the horizontal and the vertical, of the three dimensions perceived through a three-dimensional laser scan. The dimension of depth is missing from the video. In sum, different robotic systems operate with different conceptual representations of space. Sometimes only one level is used in isolation, at other times these are layered on each other.

Linguistic representations are layered too. They may be individual words, syntactic structures (sentences) or elements of discourse (utterances in a dialogue). Different approaches in the literature concentrate on different levels of linguistic description. In the following pages we review some of these approaches and discuss their relevance for our own work. We start with the discussion of the concept known as *grounding*.

2.3.1 Grounding

Connecting symbolic descriptions to non-symbolic properties of the physical world is known as *grounding* (Harnad, 1990). It is accomplished bottom up from sensory projections which Harnad calls *iconic representations* through *categorial representations* to *symbolic representations*. Categorial representations consist of selections of features that are invariant between the sensory representations of objects or events with which they can be identified. They are still non-symbolic representations (Harnad, 1990, page 342). They become symbolic when they are attached a name and are used as elementary symbols in syntactic compositions which make propositions. The compositions inherit grounding from the elementary symbols which means that all linguistic representations are deeply embedded in the physical world.

A practical challenge for a system using grounded representations is how to combine information from different modalities to achieve a responsive behaviour. Horswill (1998) describes an architecture which represents sensory data and the attentional state of the robot with a subset of modal logic with indexicals. This representation allows a very efficient implementation on hardware. It was implemented on Kludge, a robot that responds to natural language requests to fetch and follow objects.

(Roy, 2005) proposes a knowledge representation according to which grounded representations or beliefs are made part of *semiotic schemas* which are represented as structured network projections. In the schemas the beliefs are linked to other primitives: *sensors* and *actions* which connect them to the physical world, *transformers* and *categorisers* which change the knowledge contained by the beliefs, *intentional projections*

which mark beliefs to be interpreted relative to speech acts, and *generators* which turn categorial beliefs to analogue signs as in speech synthesis.

For example, objects are represented as schemas which encode their affordances and their properties. The affordances are actions that can be performed with them. In the case of a robotic arm (Roy et al., 2004) they can be detected visually or by touch. Their properties are encoded as expected values from sensors (Roy, 2005, Figure 12, page 190). The agent believes that it found an instance of an object at some location if such schema can be verified against that location. Similar schemas are also introduced for describing scenes containing objects or for describing relations between them (page 192). Schemas may also contain guides to control action which allow agents to pursue goals. This is important for representing speech acts of utterances such as "Pick up the blue one" (pages 196–198). The definition of grounding reflected in the semiotic schemas is therefore broader than in (Harnad, 1990). Grounding is not only a causal mapping between a referent and a belief, for example a cup, but also involves a prediction of agent's actions in respect to that cup (Roy, 2005, page 177).

The semiotic schemas combine information from different modalities at a single level and in a common representational form. This is in contrast to other integrated robotic systems where multi-level representations are frequently proposed (see Zender et al., 2008). The information is represented with a set of seven primitives which simplifies interfacing different modalities. Both motor actions and speech acts are represented in the same way, as actions in pursuit of some goal. This means that a planning mechanism can refer to them uniformly. Some of these ideas can be found in our own work. For example, schemas resemble the way *pDescriber* and *pDialogue* generate and interpret utterances and motion (Chapter 4). In particular, the schemas have a correspondence in the *pDialogue's* response modules which model speech acts. They contain a sequence of checks, categorisations and actions to provide a response to a user's question.

Both (Harnad, 1990) and (Roy, 2005) agree that categorisers which form categorial representations or beliefs are learned. The former assigns this task to the connectionist framework and the latter to machine learning. Roy (2002) describes how lexical words, word categorial information and syntactic structure can be learned from human descriptions of computer generated images of two-dimensional scenes containing rectangles of varying size and colour. The DESCRIBER system (Roy, 2002) starts with a sequence of words in natural language and a vector of real-valued features representing the scene in the image. Descriptions are generated by a human describer and are classified as simple "the brightest green rectangle" or complex "the yellow rectangle to the left of the large green square" depending on whether they contain exactly one object or not. All words in a description are taken as a bag of words which means that every word from the sequence could be associated with any of the features. The features referring to rectangles include the red, green and blue component of the RGB colour, their height to width ratio, surface area, the x and y position of their upper left corner and a ratio between their maximum and minimum dimensions.

The learning procedure is performed separately for simple and complex phrases. For each type of phrase it is accomplished in stages. In the first stage the classes of words are learned. The intuition behind learning word classes is that words belonging to one class have similar syntactic and semantic properties. For example, words that co-occur in the same utterance most likely belong to different classes. Also, words belonging to one class are grounded in a similar set of features. Both principles are implemented in a tailored clustering algorithm (Roy, 2002, pages 360–366). In the second stage visual features are associated with words (pages 366–367). The features of a word class are taken as the union of the features of its members. For each word a multi-variate Gaussian distribution of the subset of features selected for the word's class is estimated. This model is used to ground the word in the physical world. In the third stage, the word order of simple phrases is modelled as a statistical bigram

model based on the previously determined word classes ($P(C_{i+1}|C_i)$). A set of bigrams is represented as a statistical finite state automata (FSA). The statistical models can be used to generate the best referring expressions for a pre-specified object in the scene. The generation takes into account syntactic (order of categories), semantic (selection of words for categories) and contextual constraints. The latter are used to pick out a generated phrase that maximally distinguishes the selected object from the other objects in the scene.

While simple phrases refer to objects, complex phrases refer to relations between two objects. The FSA for simple phrases is integrated with the FSA for complex phrases and the transition probabilities between the networks are tweaked so that the parser based on the Viterbi algorithm prefers to stay within the network for a simple phrase. This adds the notion of phrase structure and recursion to the learning procedure. The reference object and the located object can be grounded from the description (page 375) and using the probabilities of the combined FSAs and some pre-specified threshold, structures such as "LO_PHRASE to the left of REFO_PHRASE"[7] can be tokenized. To learn the meaning (a multi-dimensional Gaussian distribution of features) of the remaining words which represent spatial relations, Roy uses the features from (Regier, 1996): the *proximal distance*, *proximal orientation* and *center-of-mass orientation* between the reference and the located object. The learning process includes the same sequence of procedures as for simple phrases.

As before, the generation of descriptions is driven by the principle that they must maximally distinguish the object that they ground (LO) from other objects in the scene. First, the system determines if a simple or complex description is required to ground the LO. Then, it selects REFOs that can be grounded unambiguously enough with a simple description. Finally, the best fitting spatial expression is chosen to relate the LO and each of the candidate REFOs. The REFO in the syntactic structure that max-

[7]Note that Roy (2002) uses the terms *target* and *landmark* respectively.

imises the probability according to the FSA is chosen. The sequences of words, the LO phrase, the spatial phrase and the REFO phrase are combined using the FSA for complex phrases.

Learning grounded semantic representations is also the focus of our own work. To achieve this we also use machine learning, yet in our case the learning is accomplished by general purpose classifiers rather than algorithms that are tailored for the task. Even though the approach in (Roy, 2002) is striving for automatic assignment of features to particular linguistic structures, this is not entirely the case. For example, the features involved in learning of simple phrases describing objects are not involved in learning the phrases describing object relations and vice versa. The latter are also very specific features that have been shown to be relevant for the semantics of object relations through the experimental work of Regier (1996). The syntactic structure is learned in two stages separating simple and complex phrases to avoid the difficulty of learning recursive structures. Thus, importantly, the approach is, just as ours, a supervised machine learning approach since the learning process has been guided by knowledge pre-specified by a human. There is no real measure to evaluate how much learning supervision was included in each case.

The approach uses computer generated scenes containing rectangular objects whereas our approach uses low-level spatial representations of real life scenes. In the case of the generated scenes the number of features can be enriched or reduced as desired. However, the features in our system are limited by the number of sensory subsystems that are employed on the robot. Each of these subsystems is complex and requires a research project on its own (see the CoSy project below). For example, object shape is important for the interpretation of spatial relations but sadly such a recognition module was not available on the robotic system that we used. Finally, the choice between a generated scene or a real scene also influences the accuracy of the collected linguistic information. In the generated scene the experiment designer can strictly control the

reference frame and the way the observer sees the scenes since the scene is already abstracted. In a real environment we rely on each individual informant to derive such an abstraction of the scene on their own.

The work following (Roy, 2002) extends the learning of grounded representations to real life scenes. Gorniak and Roy (2004) describe grounding of descriptions in computer generated three dimensional scenes containing cones. Although the scenes have been generated, the visual information for grounding is not based on the knowledge that generated them. Instead it is extracted from the images themselves by a synthetic vision component. Roy et al. (2004) describe a system that drives a robotic arm which can manipulate objects on a table top and uses an active vision component which generates full 3-dimensional representations of objects from 2-dimensional camera images. In both cases the learning of grounded descriptions is accomplished as in (Roy, 2002). In addition Gorniak and Roy (2004) build on the syntactic aspect of grounding and propose a *grounded semantic composition* to generate best referring expressions of the scene, and Roy et al. (2004) propose a *mental model* for grounded representations (see also Mavridis and Roy, 2006). This is a knowledge representation which allows the robot to generate imagined views. These can be used to change the frame of reference while generating object descriptions or to refer to objects currently not in view.

Grounding at the level of discourse, the highest level of linguistic representation, is described in (Steels and Baillie, 2003). They argue that grounding of natural language expressions is a shared activity and can be established through language games. Each agent in communication has access to its own internal state and representation of environment obtained through its perception. An agent describes to another agent some event in the world. The hearer can verify whether the description is true against its own internal knowledge. The communication between the speaker and hearer is open ended. The speaker can invent new meanings or new expressions which are subsequently negotiated with the hearer. This is possible because the descriptions are

grounded.

2.3.2 Instruction-based learning (IBL)

Lauria et al. (2001) and Lauria et al. (2002a) present a method where mobile robots learn from natural language descriptions and their perception of environment through instruction. The setting for the learning experiments is a miniature town (170×120 cm) which consists of an arrangement of buildings and roads closely resembling a real town. A small mobile robot (8×8 cm) is able to navigate in this town by relying on natural language instructions from its human conversational partner. The instructions are in the form of route descriptions and they can vary in complexity, depending on the knowledge of the robot. For example, in the first dialogue in Figure 2.5 the human must give the robot detailed instructions on how to reach the post office. The robot represents the knowledge contained in a description of a sequence of action chunks as a description of a new complex action which can be reused in the dialogue with a human as shown in the second dialogue. This way the robot can execute increasingly more complex tasks.

H:	Go to the post office!	H:	go to the post office at the post office turn left take a right at the crossroads tescos is on the left hand side of the street
R:	How do I get to the post office?		
H:	Er head to the end of the street. Turn left. Take the first left. Er go down the road past the first right and it's the next building on your right.		

Figure 2.5: Dialogue in IBL taken from Lauria et al. (2002b)

Not all actions are learned though. The robot starts with a set of primitive actions such as 'follow the road' and 'turn left' (for a list see Lauria et al., 2001, Table 1, page 41) each of which has been manually associated with sensory motor procedures. The set of primitive actions has been determined through experiments in which human participants generated route descriptions with no restrictions on the words they could use.

From this corpus a set 14 primitive actions was identified.[8] Each action is conceived as a triplet $S_iA_{ij}Sj$ where S_i is the pre-condition state for the action A_{ij} and S_j is the state resulting from applying the action A_{ij} to S_i. In a sequence of actions the final state of the first action must match the precondition of the next action. This way the system can verify whether a particular complex action can be executed before executing it. If not, it can query the user for more information.

The system works with two main components: a dialogue manager and a robot manager. The dialogue manager acts as an interface between the robot manager and the user. The system interacts with users through speech recognition and speech synthesis. The acoustic signals from speech recognition are analysed, parsed and represented as logical forms. The logical forms are turned into underspecified discourse representation structures (UDRSs) and later, when the ambiguities from the context are resolved, to full DRSs (Kamp and Reyle, 1993). The dialogue is modelled as a sequence of dialogue moves such as acknowledgements and questions which are represented within the same DRSs. Both the ambiguity resolution and dialogue management require inference which is provided by a theorem prover. The DRSs corresponding to the robot's response are turned into strings and pronounced by a speech synthesiser.

The robot manager interprets the DRSs from the dialogue manager and controls the robot through three main processes: execute, learn and stop. It also maintains a database of all action procedures which are represented as Python code. During the learning phase of an a new action procedure, the robot manager identifies commands and explanations within a given DRS. Actions and their arguments are extracted from them and for each action a check is made if a corresponding action procedure already exists in the database. If it does, the call to that procedure is added to the code for the new action that is being built. If not, the robot does not understand the instruction,

[8]This categorisation evolved throughout the project. For example, Kyriacou et al. (2005) give a modified list of 13 primitive actions.

the learning is put on hold and the robot manager triggers the dialogue manager to resolve the missing information.

The intuition behind IBL is that robots used in domestic environments must be able to adapt to humans. Since most of their users will not be expert roboticists who are skilled in programming the robot, the adaption must be accomplished in a way that is natural to humans: through dialogue in natural language. The system mediates between natural language descriptions and Python programming code that drives the robot. Thus, what it does is grounding but at a higher structural level than the grounding described in (Roy, 2002) and our own work. The system starts with primitive actions as innate procedures. These resemble semiotic schemas of Roy (2005) and therefore could be learned along the same lines. Finally, the robot does not use an independent planning component as in the CoSy system described below. Instead, it relies on a human conversational partner to communicate the plan.

2.3.3 Embodied robotic systems

One of the most recent embodied robotic systems which can interact with its environment and humans in natural, human-like way was developed within the Cognitive Systems for Cognitive Assistants (CoSy) project[9]. The linguistic part of research (Kruijff et al., 2007; Zender et al., 2008) was carried out at the Language Technology Lab of the German Research Center for Artificial Intelligence (DFKI) in Saarbrücken. The hardware, the system design and the environment that the robot interacts with is very similar to ours. The group works with an ActivMedia PeopleBot mobile robot which is equipped with a SICK laser[10] and a pan-tilt-zoom camera. Wheels are the sole actuators of the robot. The robot is equipped with a speaker which is connected to a speech synthesiser and humans can communicate to it using a headset connected to a

[9]The project was active between September 2004 and August 2008. Its webpage can be found at *http://cognitivesystems.org*. Its successor CogX (Cognitive Systems that Self-Understand and Self-Extend, *http://cogx.eu*) started in May 2008 and it will complete in July 2012.

[10]SICK is a company based in Germany specialising in industrial sensors.

speech recognition system. The software controlling the robot is run on the on-board computer and various other machines, all of which are connected through wireless networking. The environment with which the robot interacts is the office environment of the DFKI building.

The project concentrates on the perception of larger environments such as parts of buildings containing corridors and offices. These areas also contain objects such as sofas, desks and coffee machines. In terms of language, the research concentrates on a situated dialogue between a human instructor and a mobile robot. The human instructor takes the robot on a guided tour around the building during which the robot builds up a multi-modal spatial representation of the environment both through its sensors and through a situated dialogue with a user. The representations are built up on-line and incrementally, as the robot explores the environment.

The perception part of the system includes a SLAM component, place classification, object recognition and people tracking. The SLAM component (Folkesson et al., 2005) builds a geometric representation of the environment which is used for localising and navigating the robot. The features extracted by the SLAM are lines which correspond to walls. The place classification (Stachniss et al., 2005) uses the data from laser scans to classify locations as doorways, corridors or rooms. Doorways indicate transitions between different regions that a robot can pass through, corridors and rooms are distinguished by their geometrical features. The classifiers are trained offline through supervised machine learning. The object recognition component recognises objects from images captured by the pan-tilt camera. Again, the instances of objects are learned offline. Since the quality of images from the built in camera is quite low (320×240 pixels), only larger objects can be recognised this way. To solve this problem, a two stage process is proposed. During the first stage the location of objects in the scene is detected, in the second, the camera zooms in to that location and the objects are identified (see Zender et al., 2008, page 495ff.). The people tracking component uses

laser scan data to recognise dynamic objects. The location of the human tutor is mediated to the navigation component so that the robot follows the user but also keeps an acceptable distance.

The natural language part of the system (Krujiff et al., 2007) includes components for speech recognition and synthesis, utterance parsing and generation and dialogue. The output text from the speech recognition is parsed with a Combinatory Categorial Grammar (CCG) parser. The parses are subsequently represented as logical forms of Hybrid Logics Dependency Semantics (HLDS) (Baldridge and Krujiff, 2002). In this formalism, the logical forms consist of elementary predicates, sets of semantic features, which introduce discourse referents or define relations between the referents. More importantly, the logical forms are assigned ontological sorts which indicate, for example, the type of the propositional content specified by them (object/endurant or movement), or the intention of content toward other modalities (question, assertion and command). The information from logical forms must be interpreted against the situated context, within the context of the dialogue and in relation to different modalities (Krujiff et al., 2007, page 128ff.).

Each of the modalities produces a different representation of space. The representations are layered on top of each other and are increasingly more abstract. For example, at the lowest level there is a SLAM map which consists of lines representing walls. The next layer is a navigation graph which comprises of connected navigation nodes that are created each time the robot passes a certain distance. Navigation nodes are also assigned semantic labels (doorways, corridors or rooms) by the place classification component and also contain information about the near-by objects identified by the object-recognition component. A further layer is a topological map which is created by combining navigation nodes with the same label to areas which are separated by nodes labelled as doors. Finally, the highest level of abstraction is the conceptual map. This is partly pre-specified (Zender et al., 2008, Figure 9, page 498) and partly acquired. The

pre-specified part is the ontology of indoor office environment containing *is-a* and *has-a* relations. The first define semantic specialisation of areas such as *area.room.kitchen*, the second define containment of objects by these areas. During the robot's operation, the ontology is further specialised with knowledge that has been acquired through sensors (area1 = room) or asserted through dialogue ("This is the kitchen"). Furthermore, the reasoner can use the ontology to infer new knowledge (for details see Zender et al., 2008, page 498 and Krujiff et al., 2007, page 134).

In sum, the system builds and uses representations of space at different layers which are created by different modalities. The system is driven by natural language interpretation of dialogue. The information between the modalities is *mediated* by the BDI (Beliefs, Desires and Intentions) component of the dialogue system (Krujiff et al., 2007, Figure 6, page 134). A human instructor can issue the robot commands such as "Come with me" to start exploring the environment and "Have a look around" to start object identification. The robot can extract semantic information from sentences such as "This is the office" and "This is a bookcase" to update its ontological knowledge. This task is known as human augmented mapping (HAM). The robot can also reason using the ontology to answer questions about locations and objects such as "Where is the bookcase?". The linguistic interaction may be initiated by the robot. It can generate clarification questions to resolve uncertainties, for example "Is there a door here" to resolve whether a narrow opening in the SLAM map can be classified as a door (Kruijff et al., 2006). Finally, Zender and Kruijff (2007) describe how the ontology can be used with the algorithm of Dale and Reiter (1995) to generate contextually most appropriate description of location, for example "the hall" if both the robot and the area referred to are on the same floor, or "the hall on the second floor" if the robot is on the first floor.

Compared to our work, the CoSy project concentrates on a different level of linguistic representation: a dialogue which involves descriptions of room locations and containment of objects within the rooms rather than semantics of words referring to

robotic motion and relations between objects. The dialogue operates at the level of logical form and abstracted representations of space. Our work is similar to the place classification module which produces one type of such representations: it links the information from the low level robotic representations to the individual words using machine learning. The CoSy project does not contain a module which learns semantics of the linguistic descriptions which we are proposing to learn. Due to the similarity of architectures, our representations could be integrated with the spatial and conceptual representations of the CoSy system which would extend its linguistic competence.

2.4 Our tools

The preceding discussion demonstrates the dependence of a natural language component of a robotic system on other modalities to which it must link. In this section we describe the robotic system used in our work and the modalities that it provides. I have been privileged to use the equipment and software belonging to the Mobile Robotics Group (MRG)[11] at the Department of Engineering Science, University of Oxford. The group is led by Paul M. Newman who is also the author of the software called MOOS. The MOOS is *middle-ware* that mediates between the modalities that run a mobile robot. Its particular design allows a simplified integration of our language modules with other modalities, most of which were written by the same author as well.

We intend to use machine learning to discover relations between the structures produced by these modalities and words from linguistic descriptions. There are different ways in which such relations can be induced. We use a set of classification learning algorithms. Thus we also briefly outline the main concepts behind this method.

2.4.1 The MOOS

MOOS stands for Mission Oriented Operating Suite (Newman, 2006) and its core represents a set of libraries and executables that run a mobile robot. Its development was

[11] Webpage: *http://www.robots.ox.ac.uk/∼mobile*.

started by Paul M. Newman at the Department of Ocean Engineering at MIT and was later continued at the Department of Engineering Science at Oxford University. Recently, the core system has been made open source with intention to promote its use and development within the mobile robotics community.[12]

Its main advantages are that the system is platform independent – it can run on various flavours of Unix and Windows – and that it is conceived as a modular system with a star-like topology as shown in Figure 2.6. Each process in the figure is independent of other processes and provides a certain function. This means that components written by different researchers can be added to the "MOOS community" each of which may reflect an individual coding style of the author, may have a different internal structure or may even be written in a different programming language altogether. Secondly, since the components do not interact with each other directly, it means that they can be developed and changed independently of one another and authors can maintain their own code bases. The independence of components also means that they will be more stable and that potential difficulties with one component will not have an impact on another (Newman, 2006, pages 6–7).

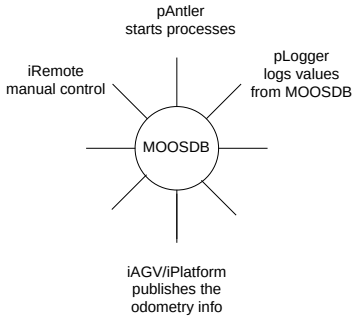


Figure 2.6: MOOS topology in general

The core of the MOOS system is a MOOS database (MOOSDB) which serves as a central repository of information, a kind of a blackboard. Communication can only be

[12]It is released under the GPL2 and it can be obtained at svn://login2.robots.ox.ac.uk/MOOS/trunk. The repository also contains the latest version of (Newman, 2006).

established from the client to the database. It is never established from one client to another or from the database to the client. One client has no knowledge of another. However, the database has the knowledge of the currently connected clients each of which requires a unique name.[13] The communication between a client and the database is in the form of messages. Each message contains information about the name of the data (a MOOS variable), its value, the name of the client sending the data, the time of the data at which it is valid, the type of the data (either string or double) and the message type.[14] Other than this, there are no additional restrictions on the data that is sent to the database. Thus, any client can publish data under any name. Furthermore, unlike a blackboard, the database not only holds the most recent value of the variable but also its history.

From the client side, messages can be of three types (Newman, 2006, page 12ff.). They can be *notifications* to the database about the value of some variable; they can be *registrations* or *subscriptions* to receive notifications which are typically published by other clients; or they can be *incoming notifications* from the database that have been requested. When a client issues one of the first two messages these are not sent to the MOOSDB straight away. The communications API on the client puts them into an outbox. Messages in the outbox are grouped together in a single packet and transmitted to the MOOS database at a pre-specified frequency such as 10Hz. Immediately afterwards, the database returns another packet containing messages that are intended for that client.[15] The messages from the packet are put into the inbox of the client where they can be retrieved by the client's collection method.

An important consequence of this information exchange is that a change in infor-

[13] There is a naming convention whereby each process name is prefixed by a letter which indicates its function. Thus, *i*Name indicates interface processes (*iRemote*, *iVoice*), *p*Name indicates pure applications that do all the work and only interact with other applications (*pSMSLAM*, *pLogger*), and finally *u*Name indicates utility applications which are not required for the system to run but may be useful otherwise (*uSMSView*).

[14] More recent versions of MOOS allow interaction of multiple MOOS communities and hence the name of the source community is also sent (see Newman, 2006, page 10).

[15] A special NULL message is created if there is nothing to transmit.

mation in the database does not trigger the information to be transferred to the client automatically. Instead, the following mechanism is implemented (Newman, 2006, pages 14–15). When a client registers to receive notifications it also requests a rate at which it wants to receive them (R_r). Another client publishes the requested information at a different rate (R_p). A MOOS client and the database exchange information at the rate R_e. Since the database keeps a history of variable values, it can return to the client upon connection a set of values not just the last one. The number of returned messages corresponds to $n = floor(R_e/R)$ where R is either R_r or R_p, whichever is the greatest. The mechanism for the information exchange is thus driven by change in time rather than change in the variable value. It follows, that a series of messages, each with a different time stamp, may be returned but all of which contain an identical value for the requested variable. Notifications thus contain sampled descriptions of states valid at some time.

Apart from the common communication framework the system also provides a set of processes that are useful in experimental work. One of the most important ones is *pLogger* which we use in our work extensively. *pLogger* can be configured to log any variable from the MOOS database at some pre-specified interval. It writes the data to two kinds of logs both of which are stored in plain text files from which information can be extracted later. Each line of a *synchronous* log contains the values of all variables that are being logged at some temporal interval. Their order is defined by their columns. The values can only be numeric (double) and therefore less useful for our work. In *asynchronous* logs, on the other hand, the value of each variable is recorded in a separate line and it may be either string or double. Depending on the interval set, an asynchronous log can record every change of a variable in the database. Entries are only made to the log once, when the variable is written. An interesting feature of asynchronous log files is that they can be played back using the *uPlayback* tool. This is useful when we want to simulate data from sensors during design of applications. In

this case we only select playback of variables published from sensors. Alternatively, entire sessions can also be played back, for example for a qualitative evaluation or a demonstration of the system.

Another useful tool is *uMS* which displays the current state of the MOOS database as a table. We can examine which processes are writing what variables at what frequencies and also their current values. Furthermore, the tool also allows us to change the value of any variable by entering it through a computer keyboard. This is useful during development of applications.

Finally, the last tool central to the operation of the system is *pAntler*. Its sole purpose is to launch other processes that constitute the current MOOS community. When run it expects a *.moos* file where these processes must be listed in its own configuration block. However, *.moos* files also contain configuration blocks of other processes in the community and *pAntler* passes the name of the *.moos* file to each of the processes launched. A *.moos* file thus encapsulates configuration of the entire community which becomes important later when data is analysed and we must refer to the conditions under which it was collected. In fact, a copy of a *.moos* file is automatically stored with a set of log files created by *pLogger*.

The integration and interoperability of systems built by different researchers presents one of the greatest challenges for practical research work, especially if this, as in our case, spans different fields. The MOOS provides one of the best platforms for research in mobile robotics in terms of system integration, stability and the ease at which information can be exchanged. Of course, as any system it comes with a few limitations which we will discuss as we describe our own contribution to it.

2.4.2 The world as seen by a mobile robot

The MOOS available under the GPL does not provide components that interact with hardware or include software related to a particular field of research. It is meant to be

a general framework to run software for mobile robots. The robots owned by the Mobile robotics Group at University of Oxford are ATRV-JR designed by iRobot who are well known for their research, industrial and military robots. In brief, a robot consists of an on-board computer with standard communication interfaces such as com ports, ethernet and wifi interfaces running a distribution of the Linux operating system. The basic machines have been further customised by the MRG with various sensors. In particular the robot that we have been using called Marge has a fixed 2-dimensional and a nodding 3-dimensional SICK laser, a sonar, a GPS receiver and wheel encoders to monitor the path and orientation of wheels. The machines are powered by a battery which provides enough power for a few hours of operation and are rugged enough to be used outdoors.

Figure 2.7: The iRobot ATRV-JR robots used by the MRG group

In addition to the hardware our work also uses software that runs it. This was chiefly designed by Paul M. Newman and is used for projects within the group. This internal branch of MOOS is known as OxMOOS. In the following two sub-sections we describe the two applications that we use in our work which provide us information about the state of the robot and the environment.

2.4.2.1 Motion of a robot

In MOOS the motion properties of a robot are commonly referred to as its *odometry*. Usually, the term "odometry" specifically refers to a distance that a body has travelled and which is measured by an odometer. However, in the MOOS system this conventional meaning is extended to include properties such as the following:

x : the current x coordinate of the robot measured in m relative to its origin (the location where the robot started when the system was turned on);

y : the current y coordinate of the robot pointing toward its front, measured in m and relative to its origin;

h : the current heading of the robot in *rad* relative to its heading at its origin: 0 degrees indicates north, negative headings indicate the robot is turned to the right and positive that it is turned to the left – this is because h really indicates *yaw*;

vx : the current velocity of the robot in the global x direction measured in m/s;

vy : the current velocity of the robot in the global y direction measured in m/s;

vh : the current angular velocity of the robot relative to its heading at its origin in rad/s;

$speed$: the current speed of the vehicle calculated from vx and vy by Pythagoras $\sqrt{vx^2 + vy^2}$ which therefore only gives positive values.

All this information is considered "odometry" because it can be inferred from the same source: the steering and motion of the robot's wheels as described in (Newman, 2003, Topic 6: Vehicle Models and Odometry). In the earlier versions of MOOS, the component calculating these values from the wheel encoders was known as *iAGV*. This was later redesigned and renamed to *iPlatform*. Although the value of the "ROBOT-NAME_ODOMETRY" variable is considerably different depending on which process

publishes it, the content is almost exactly the same (see Figure 2.8). *iPlatform* groups the values for x, y and h as a triple called *Pose* and the values for vx, vy and vh as a triple *Vel*.

(a) *iAGV*

```
MARGE_ODOMETRY: x=-2.982,y=1.160,h=2.398,vx=-0.008,vy=-0.009,vh=0.258, \
                speed=0.012,time=1105983174.213
```

(b) *iPlatform*

```
MARGE_ODOMETRY: Pose=[3x1]{0.0000,0.0000,0.0000},Vel=[3x1]{0.0000, \
                0.0000,0.0000},Raw=[2x1]{54.8000,13.3611}, \
                time=1151679972.51128792762756,Speed=0.00000000000000
```

Figure 2.8: The structure of the odometry variable

For linguistic descriptions of robot's motion the most informative measures are the current speed of the robot and its current angular velocity which tells us its heading. Also important is the direction in which the robot is travelling: is it going forward or reversing. This is not directly contained in the odometry information and must be determined separately by taking a dot product of the velocity vector and the direction vector derived from the angular velocity. Note that the angular velocity must be negated because it is a measure of *yaw* as discussed above. The sign of the resulting scalar tells us whether the robot is moving forward $(+)$ or reversing$(-)$. We append the sign to the value of speed.[16]

$$v_1 = [vx, vy] \tag{6}$$

$$v_2 = [sin(-vh), cos(-vh)] \tag{7}$$

$$v_1 \cdot v_2 = vx \times sin(-vh) + vy \times cos(-vh) \tag{8}$$

[16]I thank Alastair Harrison from the MRG for this calculation.

2.4.2.2 The robot within its environment

The location of the robot and other objects in the environment is determined by simultaneous localisation and map building (SLAM) (Smith and Cheeseman, 1986; Leonard and Durrant-Whythe, 1991; Dissanayake et al., 2001). Using this technique, a robot starts in an unknown location and in an unknown environment and uses its relative observations from its sensors to gradually build an absolute map of the environment which it can use, for example, for navigation. This makes the robot autonomous since none of the knowledge about the environment needs to be pre-specified. The technique is particularly important for navigation in environments for which it is impossible to provide a precise global map in advance. This includes most real environments, but in particular the research is important for exploration of hazardous environments such as oceans and space.

To give a rough illustration of the problem in general, imagine a situation in which person A takes a few photographs of a 3 dimensional scene such that each photograph is made from a slightly different location and captures a different part of the scene. He gives these photographs to person B who is unfamiliar with the scene and has to use some inference mechanism to create her mental representation of the scene. She may notice that a particular landmark in the scene appears in more than one photograph and thus the photographs may depict a continuous section of the scene. Even for a human this is not an easy task. The visual information provided in the photographs is not perfect: it may contain noise mostly because B is not able to infer precisely the location from which A took the photograph and because the image is distorted by the camera lens.

In a mobile robot setting (Dissanayake et al., 2001) we have a moving robot in an environment containing stationary landmarks also referred to as features. The two terms are interchangeable because landmarks are not higher level abstractions but low level representations created by sensors available to the system. In our case, the sensory

data consists of distances between the robot and landmarks which are represented as points.

The global location of the vehicle and the features is modelled as follows. The true location or the state of the vehicle at time k in the global co-ordinate frame is represented as a vector of real numbers $x_v(k)$. When the robot transitions to a new state $x_v(k+1)$ at time $k+1$, the new state is modelled as being dependent on the previous state, the transition model, the controls that moved the robot to the new state, and the process noise. Since the landmarks ($i \in N$) are stationary, their state at $k+1$ remains the same or $x_i(k+1) = x_i(k) = x_i$. Therefore, in the global co-ordinate frame all states at time k are represented as a vector $x_k = [x_v, x_1 \dots x_N]$. When a robot makes a relative observation z of some landmark i, this is related to the vector of states x_k, in particular to its items x_v and x_i, by some observation model. The observation is also affected by the noise with which it made.

Using these process and observation models the task of SLAM is to find an estimate of the location of the robot and the features in the global reference frame at time k by relying on the estimate of the state at $k-1$ and the observations as shown diagrammatically in Figure 2.9 adapted from (Dissanayake et al., 2001, Figure 1). The estimate is a probability distribution with mean $[\hat{x}_v(k), \hat{x}_1(k) \dots \hat{x}_n(k)]$ and covariance $P(k)$. It is determined recursively by a technique known as the Kalman filter. The algorithm proceeds in three phases: *Predict*, *Observe* and *Update*.

During the *Predict* phase it uses the process model and the final estimate of the state from the previous time step, $\hat{x}(k-1|k-1)$ and $P(k-1|k-1)$, to generate the first estimate of the state at the current time step, $\hat{x}(k|k-1)$ and $P(k|k-1)$. It also generates a prediction of the observation for the iTe landmark for the current time step $\hat{z}_i(k|k-1)$. In the *Observe* phase an observation $z_i(k)$ of the ith landmark is recorded. This observation contains information about the true state of the robot and the landmarks $x(k)$. A difference between the true observation $z_i(k)$ and the predicted observation

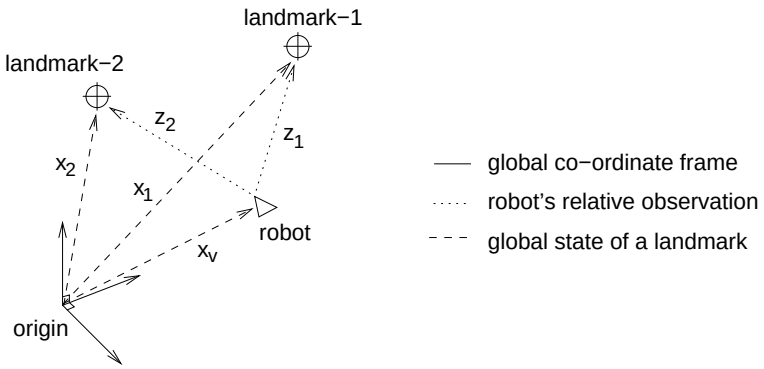


Figure 2.9: SLAM: building a global map from relative observations

$\hat{z}i_(k)$ is calculated. This difference is known as *innovation* $\tilde{y}_i(k)$ with covariance matrix $S_i(k)$. In the third phase called *Update* the innovation $\tilde{y}_i(k)$ is used to improve the state estimate and the associated covariance giving us the estimates $\hat{x}(k|k)$ and $P(k|k)$.

Dissanayake et al. (2001) show proofs based on the properties of the state covariance matrix $P(k|k)$ that as the robot is moving and making more observations under the above model, the map that the algorithm builds, represented by the relationships between the landmarks, converges. Firstly, they show that as successive observations are made the uncertainty of the state estimates reduces monotonically. When the number of observations approaches infinity the uncertainties of the relative locations of landmarks converge to zero. This means that with successive observations the states of landmarks become increasingly correlated and when the limit is reached they become fully correlated. Thus, if we know the location of one landmark with some uncertainty we can determine the location of another one with an identical uncertainty. Finally, they prove that with increasing observations the error of absolute location of every landmark on the map converges to the error equal to that with which the first observation is made.

The algorithm outlined here demonstrates the basic concept of SLAM. However, in practical applications several extensions to it have been introduced (Newman and

Durrant-Whyte, 2001; Newman and Leonard, 2003). The OxMOOS implements a pose based formulation of the SLAM algorithm based on Bosse and Zlot (2008) in the process known as *pSMSLAM*. In contrast to the SLAM component used in the CoSy project, *pSMSLAM* produces a map which is a set of n-tuples $\langle x, y \rangle$ or $\langle x, y, z \rangle$ depending whether a 2-dimensional or 3-dimensional laser is used. The x, y and z coordinates are expressed relative to the origin of the robot which is the point in the environment where the robot started. The map can be viewed graphically using the *uSMSView* tool as shown in Figure 2.10.

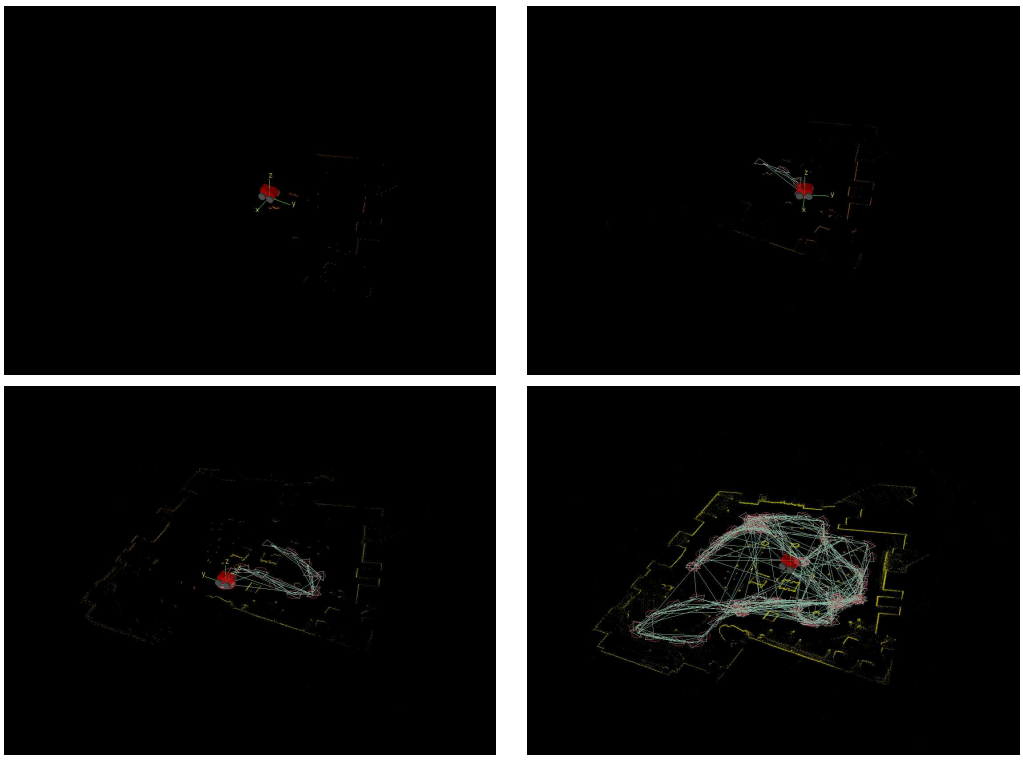


Figure 2.10: Building a SLAM map with MOOS. We can see a series of screen-shots from the MOOS utility *uSMSView* taken while the robot "explored" the environment and incrementally built a map. Although objects can be quite easily discernible to a human eye it is important to note that for the system they are only sets of points in a 2-dimensional coordinate space.

pSMSLAM can be used in two modes of operation. In the SLAM mode the module builds a map incrementally from its observations as it moves around the room and localises the robot on the map as shown in Figure 2.10. The map can be saved to a file

and reloaded to *pSMSLAM* when run in the localisation mode. In this mode the system tries to match its current observations from laser scans with the map that was supplied to it. If the system can determine the current location of the robot on the map it also means that it has knowledge of the location of all other points in the environment. The localisation mode is particularly useful in experimental work since this eliminates the need to rebuild the map between different runs of the system. Of course, the map can only be reused if the scene has not changed in the meantime.

Objects appear to the system only as clouds of points rather than discrete entities and we have to identify them manually. A small addition to the *uSMSView* was implemented which gives us, for every point that we choose on the graphical map, the coordinates $\langle x, y, z \rangle$ relative to origin of the robot. We represent every object as a point defined by its central axis and a name as shown in Figure 2.11. Such a representation ignores the shape of objects: it is suitable for objects that are round or square but less suitable for objects that are long and thin. We chose this representation to keep the features for learning as simple as possible with the fewest abstractions.

Figure 2.11: Objects were grounded as points of their central axis

The names of objects and their absolute locations are not saved with the SLAM

map. Instead they are a part of the configuration of the *iCommentary* application[17] which monitors the location of the named points and publishes them to the MOOS database at some pre-specified interval. The information is published as a set of 4-tuples $\langle$object-name, $x, y, z\rangle$ under two different variables. "COMMENTARY_LOCATIONS" contains 4-tuples with absolute coordinates of object locations which are static and are just replications of the coordinates which we identified the objects on the map with. "COMMENTARY_RELATIONS" on the other hand contains 4-tuples with coordinates of object locations relative to the current location of the robot.

The model of projective spatial relations (Maillat, 2003) discussed in Section 2.2, page 12 proposes that the origin of the coordinate system is always fixed by the reference object regardless whether intrinsic, relative or absolute reference frame is intended in their interpretation. The orientation of the coordinate frame is determined by different contextual parameters (Table 2.1). Projective descriptions in English are ambiguous in their interpretation according to intrinsic and relative reference frames. Apart from projective descriptions our dataset may also contain topological descriptions such as "near" whose reference does not require a coordinate frame or the orientation parameter but only considers the distance between the two objects.

How much information from the model should be integrated in the dataset for machine learning? In order to apply the complete model, the coordinate system that defines the coordinates in "COMMENTARY_RELATIONS" would have to be transposed onto the reference object. Then, it should be determined whether a particular description of a relation is topological or projective. In the latter case the coordinate system should be rotated according to the intended viewpoint parameter which would have to be determined from the context or from the discourse. The coordinate frame of coordinates in "COMMENTARY_RELATIONS" is already orientated according to the robot and hence no rotation would be required if the relative reference frame is in-

[17] I thank Paul M. Newman for implementing this application.

tended and the descriptions are made from the robot's point of view. For the intrinsic reference frame one would have to know the orientation of the reference object which would have to be specified for each object manually. It follows that the application of a full model requires a lot of information which is simply not available from the SLAM map.

Of course, if a full model is applied, machine learning is redundant. It would only learn spatial templates which can be determined with a far more straightforward method (Logan and Sadler, 1996). On the other hand, if completely random information is given to learners, nothing useful will be learned. Therefore, a compromise was made to use the coordinates of objects without transposition as given in "COMMENTARY_RELATIONS". These express the locations of objects local to the current position of the robot without assuming a particular model. The describers were asked to make descriptions from the perspective of the robot. In most cases, when the reference object is another object in the scene, this requires the application of the relative reference frame. However, if the reference object is the robot itself, then the relative and intrinsic reference frames coincide. We expect these distinctions will be learned by the learners.

The way the robotic system perceives the environment is much different from the way humans perceive it. It uses primitive low level information which is extremely accurate. Such information is well suited to navigate and localise a mobile robot. Humans on the other hand refer to their environment using abstract concepts which are quite vague (van Deemter, 2006). The challenge of this work is to show that the low level information from the robotic system can also be used for abstracting the concepts through which humans perceive their environment. If we are successful, then it can be concluded that the robots can simulate some human behaviour.

2.4.3 Machine learning

Automatic extractions of useful information from datasets is commonly known as machine learning or data mining. However, there are a few different techniques of ma-

chine learning that are employed. They differ in the type of data that they can handle and what *generalisations* about the data are made. We have already discussed one of the techniques when discussing SLAM. *Estimation* is a sub-case of *numeric prediction* where given some attributes (the variables in the transition and observation models) a numeric value is predicted which describes the state of the robot or a feature in the global coordinate frame.[18] *Classification learning* is similar to numeric prediction except that instead of predicting a numeric value a nominal value is returned. Yet another kind of learning is *association learning*. This does not return any particular numeric or nominal value but finds any associations between the attributes that are passed to it. Finally, *clustering* finds examples that appear similar in terms of their attribute values and groups them in a cluster.

For our task numeric prediction and classification appear to be the most suitable methods. Rather than implementing the algorithms ourselves, we chose to use Weka (Witten and Frank, 2005) which stands for Waikato Environment for Knowledge Analysis and is developed at the University of Waikato. This contains a ready implementation of many popular machine learning algorithms and associated tools and is written in the Java programming language. Although some machine learning algorithms such as the Naive-Bayes are not too difficult to implement independently, the main advantage of Weka is that it provides a unified toolbox for preparing and manipulating input datasets, performing learning on them, using the learned knowledge to make new predictions on unseen datasets and evaluating the performance of the learning tasks. Weka introduces a unified format of input files known as the Attribute-Relation File Format (ARFF). Datasets written in this format can be used with any machine learning algorithm included. This makes a comparison of their performance a straightforward task. Furthermore, it also includes a learning evaluation component which provides common evaluation techniques such as stratified 10-fold cross-validation and calcu-

[18]This corresponds to the *localisation* mode of the *pSMSLAM* process which uses the map created in the *estimation* step to navigate the robot or, as in our case, to locate the objects.

lates many popular evaluation metrics such as the F-measure. Finally, because the software is written in Java it can be easily extended with new classifiers or components.

Weka algorithms cannot make generalisation from any unstructured dataset. Information must be organised into *instances*. One can think of an instance as an observation of a set of variable values. In Weka terminology variables are called *attributes* and hence an instance is a vector of attribute values. A typical example dataset used in the machine learning literature (and also in Witten and Frank, 2005) is the weather dataset which contains observations of various weather attributes and whether people played some sporting game as shown in Figure 2.12.

Outlook	*Temperature*	*Humidity*	*Windy*	*Play*
sunny	85	85	false	no
rainy	70	96	false	yes
sunny	75	70	true	yes

Figure 2.12: An extract from the weather dataset

A collection of instances can thus be represented in a table. As previously mentioned, the task of a machine learning algorithm is to uncover relations between the attribute values that correctly generalise all instances in the training set. Importantly, however, the generalisations can only be made on relations between attribute values and not across different instances. Instances are independent and represent different examples of the same generalisation or the *concept* that is learned. This contrasts to the estimation problem in SLAM where observations of the robot's motion and observations of the environmental features are also temporally dependent.

The independence of instances may sometimes be problematic. Witten and Frank (2005, page 46) give an example where an ancestral tree is conflated to a table of instances where the attributes are simply *Name*, *Gender*, *Parent1* and *Parent2*. If instances are given in this way, it would be impossible for the learning algorithms to make any

useful ancestral generalisations such as "mother of" since all of these relations are encoded between instances and not within the instances in terms of attribute values. Fortunately, there are ways to transform such datasets to a table of independent instances by introducing new attributes. However, this means that their size becomes larger which increases the cost of computation and storage. It may also create spurious relations which we are are not interested in uncovering since they reflect the structure of the original data which is already known to us. Another difficulty in representing an ancestral database in a table of independent instances is that ancestral generalisations are infinite which is something that cannot be represented in a finite table.

The information encoded in instances only tells the algorithms that a certain observation is true. It is impossible to encode knowledge that something is not true as expressed by the proposition "the outlook is not sunny". Instances can only encode positive knowledge. All instances that could have been formed from the remaining combinations of attribute values are explicitly assumed to represent negative knowledge. This is known as the *closed world assumption*.

Different machine learning methods will make different generalisations. In the case of classification the goal of a machine learning algorithm will be to predict the value of one attribute which in Weka terminology is known as the target class. Returning to the weather dataset, the target class will be the attribute *Play* since we are interested in creating a classifier that will predict whether a game is played or not given the current weather conditions. In terms of the classification algorithms there is no restriction on which attribute can be the target concept – one could also choose *Outlook* for example. However, a classifier that predicts whether it is sunny or rainy given that people are playing the game is probably not very useful to us. Sometimes very strange and unintuitive generalisations can be made. The algorithms have no knowledge which attribute is the most appropriate target class – instead this must be decided by a human. For this reason classification is known as a *supervised* method of machine learning.

Although the input knowledge to the learning algorithms is standardised in Weka, there is no standard form of the output knowledge. Instead this depends on each particular learning algorithm employed. For example, Naive-Bayes produces probabilistic knowledge, whereas a decision tree learner produces knowledge in the form of a decision tree. Of course, in both cases the knowledge is used to make a classification but the steps leading to it may be quite different.

2.5 Conclusion

This chapter gave an overview of some of the ideas, approaches and tools related to our work. We are now ready to start the discussion of how the learning of descriptions was performed and its success evaluated.

Chapter 3

Machine learning

3.1 Introduction

In this chapter we discuss how we use machine learning to learn the grounded meanings of spatial words. The process of machine learning can be divided into various stages. First, a corpus of suitable observations must be collected which serves as the training data from which generalisations are made. The observations must be structured in a particular way to form instances which can be fed to machine learning classifiers. Importantly, the observations are assigned category labels which constitute additional knowledge in the machine learning process. This is why such method is known as a supervised method. We structure instances according to our assumptions and theories about the data. There is no guarantee that these aid the discovery of the true underlying relations in the data if these exist at all. Another important factor is the algorithm used for learning. Different algorithms make abstractions about the data in different ways and thus give different results. We use two algorithms: a symbolic and a probabilistic one. In the final section we discuss the success of learning. In particular we contrast the classifier performance relative to the various choices that we took in the preceding steps.

3.2 Data collection

3.2.1 Linguistic and non-linguistic data

The objective of our machine learning experiment is to find meaningful associations between (i) odometry or topological data representing the state of the robot or the environment, and (ii) natural language data obtained from human descriptions. Each observation used for machine learning will contain a set of values consisting of either odometry information or topological information, and natural language information.

We collected two datasets of observations which were made under slightly different conditions: a small development dataset and a larger final dataset. We refer to these datasets as *Simple* and *All*. The datasets differ in regard to how linguistic information was sampled. The *Simple* dataset was created from descriptions provided by a single human describer. The describer did not have to generate full sentences, but he chose the words to create descriptions from a pre-defined list that was displayed on a computer terminal. The vocabulary was thus controlled. It only contained the most frequent descriptions of motion and object relations as summarised in Table 3.1. We grouped the words describing motion into four different categories which in our intuition correspond to human conceptual classes.

Category	Words
Verb	moving, stopped
Direction	backward, forward, none
Heading	right, left, none
Manner	fast, moderately, slowly, none
Object	box, chair, chest, desk, pillar, shelf, table, tyres, you
Relation	behind, in front of, to the right of, to the left of

Table 3.1: Controlled vocabulary used in the *Simple* dataset

The *All* dataset was collected from descriptions of four describers, two native and two non-native but proficient speakers of English. The vocabulary was not restricted in any way. The describers were instructed that they can use any descriptions of mo-

tion or location of objects that they consider natural as speakers of English. Table 3.2 shows a selection of descriptions that the describers used. In the majority of cases the describers used mono-clausal descriptions containing words belonging to the four basic categories identified in Table 3.1. There is also some variation in the structures produced. Some describers were more creative than others. They used complex descriptions ("The table is in front of you and to your left"), adverbial modification ("You are directly behind the box", "The chair is slightly to the right of you" and "You are somehow between Homer and the box") and even negation ("The shelves are not close to you"). However, such descriptions were relatively rare.

Category	Descriptions
Motion	Moving forward left.
	Reversing right.
	Stopped.
	Moving backward very slowly.
	Turning to the left.
	Turning clockwise.
	Moving forward and turning right quite slowly.
	Turning on the spot to the left.
	Very gently turning to the right and then moving forward into the right at medium pace.
Object relations	The table is to your left.
	The desk is in front of you.
	The nearest object to you is the table.
	Homer is in front and to the left of you.
	The barrier is to your left further away than the table.
	The box is directly to your left.
	The table is between you and the chest.
	The chest is to the right of the table.
	You are closer to the table than the shelves.
	Flakey is directly behind the box and to your right.

Table 3.2: Free vocabulary and sentence constructions used in the *All* dataset

As described later in Section 3.3.3, a set of words was collected from these utterances and each word was assigned one of the categories *Verb*, *Direction*, *Heading*, *Manner* and *Relation*. If a word did not fall into any of them, it was discarded. The names of the objects did not have to be tagged because their names can be extracted automat-

ically from the locational data. The data collected in the *Simple* dataset was also added to the *All* dataset. The tagging procedure resulted in the vocabulary given in Table 3.3. With the exception of object names, these are the words whose grounded meanings will be learned.[1]

Category	Words
Verb	going, edging, continuing, reversing, creeping, turning, moving, stopped
Direction	spot, backward, forward
Heading	anticlockwise, clockwise, straight ahead, hard, straight line, around, straight, 180, right, left
Manner	quickly, walking pace, imperceptible, tightly, gently, rapidly, fast, moderately, slowly
Object	barrier, bin, box, chair, chest, cupboard, desk, Flakey, Homer, origin, pillar, shelves, table, tyres, wall, you
Relation	next to, after, near, parallel to, opposite of, facing, in front of, far, to the right of, to the left of, behind, close

Table 3.3: Extracted vocabulary used in the *All* dataset

The odometry information that was used in learning includes the estimation of speed (*Speed*) and orientation of the robot (*Delta-Heading*) as discussed in Section 2.4.2.1. The topological information contains the x and y coordinates of objects in a two-dimensional coordinate space relative to the current location of the robot as discussed in Section 2.4.2.2. Descriptions of objects only refer to pairs of objects, the located and the reference object, and hence only the coordinates of these two objects are relevant for learning descriptions of relations between them. We represent them as attributes *LO_x*, *LO_y*, *REFO_x* and *REFO_y*.

The nature and the detail of odometry and topological information is the same in both datasets. However, the scale of their numerical values may be affected by the maximum speed and rotation that the robot can achieve in a given environment and the size of the room which all varied between the scenes in which the datasets were

[1]For attributes *Direction*, *Heading* and *Manner* a value "none" is also learned which corresponds to a case where no word of that category is encountered in the utterance. For more details see Section 3.3.3, page 71.

created. We return to this problem in Section 3.3.2.

In sum, our learning datasets contain eleven attributes, four linguistic (*Verb*, *Direction*, *Heading*, *Manner* and *Relation*) and six non-linguistic (*Speed*, *Delta-Heading*, *LO_x*, *LO_y*, *REFO_x* and *REFO_y*). Each learning task consists of building a classifier that predicts the values of one of these attributes by relying on the values of other attributes. Not all attributes are used in each learning task. We give an overview of input attributes for each learning task in Section 3.3.4. Because the robot is mobile and may change locations, and because each type of data is collected from a different source, it is important to ensure that the collected linguistic and non-linguistic observations are temporally synchronised before they are written as attribute values of instances. We address this issue in Section 3.3.1.

3.2.2 Experiment set up

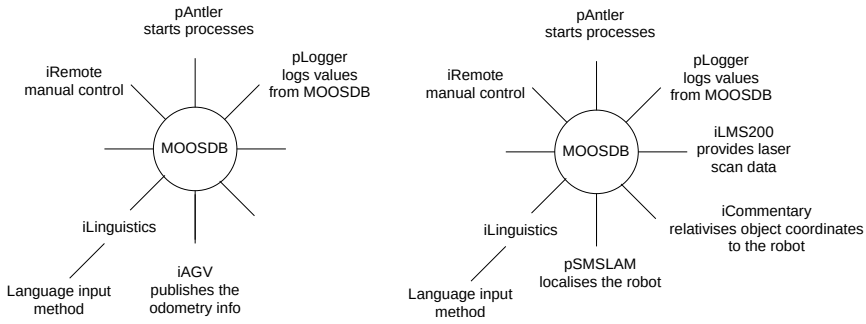
For both *Simple* and *All* datasets the descriptions of motion and object relations were collected in separate sessions. During the first session, the operator guided the robot in an enclosed space trying to display various types of motion that the robot is capable of. Human commentators were asked to describe the motion while the robot was moving. During the second session, the operator first moved the robot to a certain location in a room containing various objects and then human commentators were asked to describe the location of the robot and the location of the surrounding objects as if the descriptions were from the perspective of the robot.

The reason why the two types of descriptions were collected separately is mainly practical. In order to generate a variety of motion a considerable space is required which would be further restricted if objects were also present. It follows from this experimental setup that odometry information and information about the locations of objects will never be present together in any machine learning task. This decision is reasonable since we do not expect that properties such as the speed of the robot will

influence the choice of a description of object relation. On the other hand, we can expect that a presence of an object in some region around the robot may be related to a description such as "moving right". However, the robot's speed and heading are far more accurate sources of data from which the grounded representations of descriptions of motion can be learned. If certain properties are irrelevant for learning a certain type of description, the learning algorithms should be able to exclude the contribution of such attributes automatically as shown in the experiments of Roy (2002) discussed on page 21. However, this assumes that the descriptions are learned from a large dataset where every value of the irrelevant attribute is found with every value of the target concept at least a few times. Otherwise, spurious relations may be discovered. Inclusion of irrelevant attributes therefore increases the data requirement and if this is not satisfied it has a negative effect on the success of learning.

The system was set up slightly differently in each data collection session as shown in Figure 3.1. While collecting the motion data the operator used *iRemote* to move the robot around the room and *iAGV* provided odometry information. To ensure that each human describer would be given an opportunity to describe a broad range of motions, the operator had a list of possible motions that he referred to while guiding the robot. This list, however, was not disclosed to the describers. Roughly, the motion types included forward and backward motion with various velocities and turning left or right under acute and obtuse angles, rotating the robot on the spot and stopping. *iRemote* accepts commands from a standard computer keyboard: the values of the desired rudder and desired thrust can be incremented in steps of 2. They are percentages of the maximum rotational and transport velocities that the vehicle is allowed to achieve which in turn are measured in metres per second and radians per second respectively. The desired rudder and thrust are taken as instructions to *iAGV* of what percentage of the maximum velocities the vehicle should achieve at any given moment. The maximum velocities can be defined by the operator for every session. The typical values that we

used were 0.6 and 0.4 m/s and 0.8 and 1.0 rad/s. We varied these while collecting the *Simple* and *All* datasets to change the responsiveness of the robot. These parameters became part of the dataset: while creating instances we normalised the observed angular velocities and speed to these values.



(a) Motion data (b) Object relations data

Figure 3.1: The MOOS topology used during data collection

The linguistic data was collected using two different language input methods. While compiling the *Simple* dataset, the user was presented with a list of word choices on a terminal window. He generated scene descriptions by entering the numbers associated with each each word that best described the scene. The words were grouped into categories and the describer could only choose one word per each category. The benefit of this input method is that it restricts the describer to a small core vocabulary which we were planning to collect and that entering descriptions is very fast which was particularly important in the case of motion descriptions. Because the robot was moving in an enclosed environment it was difficult to sustain a particular motion for a prolonged period of time. If the describer was too late in generating a description, either because it took time for him to decide on the description or because he was limited by the input method or the system and the motion of the robot would have changed in the meantime, the resulting description would be incorrect. This means that considerable error could be introduced to the dataset.

When collecting the *All* dataset we did not want to limit the describers to a partic-

ular vocabulary and hence the previous input method became unsuitable. Instead we choose speech recognition[2] which has proven to be the fastest and the most accurate method of collecting linguistic data. The speech recogniser did introduce some errors by incorrectly recognising certain words, but these were consistent and could be easily corrected manually later. Using a speech recogniser also meant that each participant in the experiment had to train the recogniser first to their speech which took approximately 15 to 20 minutes. Alternatively, one section of a dataset from one describer was collected without a speech recogniser where the describer entered the descriptions by typing them on a computer keyboard. The intention of this experiment was to allow for a rough comparison of both input methods to see which one gives instances of higher quality containing fewer errors.

In the second set of sessions where we were collecting a dataset of object relation descriptions we set up an environment with real-size objects such as a chest, a box, a table, a pillar, a stack of tyres, a chair, a desk, shelves, etc. A portion of this environment including the robot is shown in Figure 3.2. The system was configured as shown in Figure 3.1b. The non-linguistic data is provided by *pSMSLAM* which is responsible for localisation and map building and *iCommentary* which makes the coordinates of objects relative to the current location of the robot. *pSMSLAM* relies on the *iLMS200* process which controls the laser and returns the scan data. As previously described, the map is built in two stages. In the first stage we set *pSMSLAM* to the SLAM mode and navigated the robot manually around the room until a map with sufficient detail was built. We identified and named the objects on the map and declared their names and their absolute coordinates in the *iCommentary* configuration block of the *.moos* file. When an experiment started, *pSMSLAM* was set to the localisation mode and was configured to use the map that it has previously built. The size and the shape of the room were different during the creation of the *Simple* and *All* datasets. We assume that the

[2]We used Dragon Naturally Speaking 8 by Nuance Communications, Inc. which we linked with our own code to the MOOS system.

size is important for the interpretation of the meaning of spatial relations. The size can be extracted from the MOOS map and therefore the map becomes a part of the dataset. We discuss the normalisation of object coordinates to the size of the environment in the following section.

Figure 3.2: The describers were asked to describe the scene from the perspective of the robot: the table is to the right of the box and the tyres are to the left of the robot.

We logged both linguistic and non-linguistic data using the standard MOOS logging tool known as *pLogger*. As described in Section 2.4.1 of the previous chapter, logs can be synchronous or asynchronous. Unfortunately, the synchronous logs can only be used to log MOOS variables that are numeric. However, none of our variables are numeric but complex string attributes that contain a list of comma separated feature-value pairs as shown in Figure 3.3. This means that only asynchronous logs could be used. The difficulty that arises with these logs is how to match their entries each of which has a different time stamp. Sometimes there are multiple qualifying entries for a certain variable as in the case of "MARGE_ODOMETRY", in other cases the difference in time stamps between the two qualifying variables is quite wide as in the case of "COMMENTARY_RELATIONS". Again, in certain very rare cases, the variable

may not be logged at all in the required time interval. We return to these issues in the following section.

```
 66.230   DESIRED_THRUST   iRemote   6
 66.230   DESIRED_RUDDER   iRemote  -4
 67.255   MARGE_ODOMETRY   iAGV   x=0.454,y=2.964,h=1.097,vx=-0.052, \
    vy=0.027,vh=0.052,speed=0.059,time=1132139182.726
 67.363   MARGE_ODOMETRY   iAGV   x=0.449,y=2.967,h=1.103,vx=-0.041, \
    vy=0.021,vh=0.041,speed=0.046,time=1132139182.834
 67.428   VOICE_INPUT   iLinguistics   moving forward slowly
 67.471   MARGE_ODOMETRY   iAGV   x=0.443,y=2.970,h=1.110,vx=-0.048, \
    vy=0.024,vh=0.059,speed=0.054,time=1132139182.942
...

627.602   COMMENTARY_RELATIONS    iCommentary    Vehicle-Wall= \
   [3x1]{-1.388,-2.974,-0.2557},Vehicle-Pillar=[3x1]{-3.694, \
   1.944,-0.2557},Vehicle-Desk=[3x1]{-0.9161,3.673,-0.2557}, \
   Vehicle-Shelves=[3x1]{3.622,3.339,-0.2557},Vehicle-Cupboard= \
   [3x1]{3.547,1.963,-0.2557},Vehicle-Homer=[3x1]{-0.8307,2.022, \
   -0.2557},Vehicle-Table=[3x1]{-1.198,0.02526,-0.2557}, \
   Vehicle-Chair=[3x1]{1.688,2.268,-0.2557},Vehicle-Box=[3x1]{1.92, \
   0.08886,-0.2557},Vehicle-Flakey=[3x1]{2.111,-1.253,-0.2557}, \
   Vehicle-Bin=[3x1]{2.029,-2.756,-0.2557},Vehicle-Barrier= \
   [3x1]{-3.158,-0.1599,-0.2557},Vehicle-Chest=[3x1]{-2.762,-1.607, \
   -0.2557},Vehicle-Tyres=[3x1]{0.5896,-1.243,-0.2557},Vehicle-Origin= \
   [3x1]{-0.3618,-1.781,-0.2557},
641.112   VOICE_INPUT   iLinguistics   the table is to your left
```

Figure 3.3: An extract from asynchronous MOOS log files. Note that odometry and locations of objects were not logged together as they were collected in separate sessions.

Before starting collecting descriptions of either type, describers were familiarised with the environment. For motion data, the operator showed them briefly what types of motion the robot is capable of. For descriptions of object relations, the operator explained to them the names of the objects in the room that they could refer to. The describers were instructed to produce descriptions from the perspective of the robot and to clarify what is meant by this, the operator gave them a few simple examples. They were asked to walk around the room to build their mental representation of space. They were also allowed to do this while making descriptions of object relations once the experiment has started as this helped them to visualise the scene better from the perspective of the robot. It soon became evident that the describers were making a considerable number of mistakes related to the perspective: they switched to their own

perspective or to the perspective of the reference object if it had an identifiable front. This presented another source of error for our dataset. Once the describers exhausted the description possibilities that could be generated from one robot location, they were instructed to inform the operator to move the robot to another randomly chosen location. The data collection experiment was not limited in time but the participants spent approximately 1.5 hours in total in the lab.

3.3 Creating instances for machine learning

The algorithms in the Weka toolkit (Witten and Frank, 2005) that we use are supervised methods of machine learning. They cannot process just bags of data of various types to induce meaningful theories. Instead they require a set of observations known as instances which are lists of values of a predefined and fixed set of attributes. Each attribute defines a category of observations. The decisions that are introduced by a human in preparing a learning dataset represent an important knowledge in the process of finding the theories that describe the data and are thus significant for the outcome of learning as already discussed on page 56. We model our data with attributes which we believe are related and the goal of learning is to find this relation. In this section we describe how the robotic and natural language data was structured and prepared for machine learning.

3.3.1 Combining log entries to instances

We created our raw datasets with *pLogger* which creates for each variable value logged a separate time-stamped entry. This means that linguistic data is logged separately from non-linguistic data. In most cases, the variable values are lists of complex feature-value pairs. To create instances containing both kinds of information we must therefore find a related set of log entries, extract the relevant attribute data from it, and rewrite the data as a single instance containing these attribute values $\langle$Val1, Val2, Val3...$\rangle$. The procedure must be accomplished automatically because of the large number of

observations. Also, non-linguistic descriptions are meaningless to humans and hence no value would be added if instances were created manually.

The way we combine log entries to instances becomes particularly important in the case of motion data because the conditions were changing fast while describers were making descriptions and it was possible that the data became unsynchronised. We used two algorithms to extract the data from logs and to create instances. We compare the performance of machine learning classifiers built from the instances created by each algorithm and discuss the differences in their performance later in this chapter.

In general we assume that a human describer makes a description after observing the environment around them. Thus, a linguistic description made at *dtime* will always temporally follow the descriptions of the environment and the state of the robot made at *ptimes*. Thus, we should look for non-linguistic descriptions that immediately precede the linguistic ones in terms of their time stamps. Figure 3.4 outlines the algorithm.[3] In item (1(a)iv), the *ptime* may not be found if *dtime* occurs at the beginning of a log file where a linguistic description may precede the logging of environment properties or in rare cases where two linguistic descriptions would follow one another with no intermediate description of the environment.

create_instances: takes an ordered list of description time codes *DTimes* and an ordered list of environment/robot property time codes *PTimes*.

1. Take *dtime* as the first member of *DTimes*.
   (a) **find_property_time:** using *dtime* and *PTimes* find *ptime*.
      i. Take *ptime* to be the first member of *PTimes*.
      ii. If $ptime \leq dtime$ and (next(*ptime*) $> dtime$ or next(*ptime*) $= \{\}$), add the pair (*dtime*, *ptime*) to the output list *DTimesPTimes* and stop.
      iii. Else recurse (1a) on the tail of *PTimes*.
      iv. Fail if $PTimes = \{\}$.
2. Recurse (1) on the tail of *DTimes* until $DTimes = \{\}$.
3. Return the list *DTimesPTimes*.

Figure 3.4: The basic algorithm for associating linguistic description time codes with environment/robot property time codes to create instances from MOOS logs.

[3]The algorithms were implemented in Prolog (Bratko, 2001) and thus we also write them in a way which resembles its coding style.

We already mentioned that we observed a notable delay from the time when a user observed the motion of the robot and until the description became logged to a file. The describer has to decide on the description, select options on the computer keyboard or generate an utterance, the utterance must be recognised by the speech recogniser, sent to the MOOS database, retrieved by *pLogger* and finally written to a log file. This delay is not constant during the session of one describer. There is also a considerable difference in the detail and richness of linguistic and non-linguistic information. Non-linguistic information is logged at small intervals at a couple of milliseconds, carefully reflecting the state of the robot and the environment. On the other hand, linguistic information is published every couple of seconds when a describer has something to say. It follows that the previous attempt to take the immediately preceding non-linguistic log entry for every linguistic entry may not be the optimal one: there may be better entries to choose from.

The only time when we can be sure that linguistic descriptions are matched with descriptions of the state of the robot in a log file is when the robot is stationary and the describers refer to its state with the word "stopped" or its variant. We collect a list of time codes of these entries as shown in Figure 3.5. To this list we also add the time code of the last entry of the log file if this is not already a description containing "stopped". To match the description "stopped" with a description of the robot's state (Figure 3.6) we use the basic matching strategy given in Figure 3.4. If both entries correspond as expected, the linguistic descriptions are not delayed and our work is done. On the other hand, if the robot is not stationary, the descriptions are delayed. The delay can be estimated by regressing on the list of time codes of odometry entries until we find an entry when the robot is stationary (Figure 3.7). The time difference between the description "stopped" and this odometry entry is taken as our estimation of delay. We use this delay to shift the time codes of all linguistic descriptions preceding the "stopped" description. The method thus splits log entries from one session to

find_stop_points: takes an ordered list of linguistic description time codes *DTimes*, a list of natural language words that describe a non-moving robot *StopWords* and an ordered list of all time codes in the log file *LTimes*.

1. Using *DTimes* and *StopWords*.
   (a) Take *dtime* as the first member of *DTimes*.
   (b) Extract a list of words *Words* from the entry with *dtime*: words(*dtime*) = *Words*.
   (c) **sentence_contains_stopped:** try to match the members of *StopWords* to *Words*.
      i. Take *stopword* as the first element of *StopWords* and check if $stopword \in Words$. If true, stop.
      ii. Else recurse (1c) on the tail of *StopWords*.
      iii. Fail if $StopWords = \{\}$.
   (d) If **sentence_contains_stopped** is true: add *dtime* to *StopTimes* and recurse (1) on the tail of *DTimes*.
   (e) If **sentence_contains_stopped** is false: recurse (1) on the tail of *DTimes*.
   (f) When $DTimes = \{\}$, return *StopTimes*.
2. Using *LTimes*.
   (a) Find $last = \text{last}(LTimes)$.
   (b) If $last \in DTimes$ Then
      i. words(*last*) = *Words*
      ii. If **sentence_contains_stopped(***StopWords*,*Words***)** is true, stop.
      iii. Else append *last* to the tail of *StopTimes*.
3. Return *StopTimes*.

Figure 3.5: Finding stop points: a stop point is either a time code of a linguistic description which contains the word "stopped" or its variant or the last entry of the log file.

segments, each of which concludes with an entry containing a linguistic description "stopped" and for each of which a different delay is calculated. Finally, we use our basic matching algorithm to match the shifted time codes of linguistic descriptions to the time codes of the odometry information.

create_segment_instances: takes an ordered list of stop point time codes *StopTimes*, an ordered list of description time codes *DTimes* and an ordered list of odometry time codes *OTimes*.

1. Using *StopTimes*, *DTimes* and *OTimes*.
   (a) Take *stoptime* as the first element of *StopTimes*.
   (b) Using *stoptime* and *OTimes* **find_delay** *delay*.
   (c) Match descriptions with odometry using *delay*, *stoptime*, *DTimes* and *OTimes*.
      i. Take *dtime* as the first element of *DTimes*.
      ii. $shifted\_dtime = dtime - delay$
      iii. For *shifted_dtime* and *OTimes* use **find_property_time** to find the immediately preceding odometry time *otime*.
      iv. Add the (*dtime*,*otime*) pair to the output list *DTimesOTimes*.
      v. Recurse (1c) on the tail of *DTimes* and *OTimes* while $dtime \leq stoptime$.
   (d) Return the *DTimesOTimes* list and the tail of *DTimes*.
2. Create instances from the *DTimesOTimes* pairs.
3. Recurse (1) on the tail of *StopTimes*, the tail of *DTimes* and *OTimes*.
4. When $StopTimes = DTimes = \emptyset$, stop.

Figure 3.6: Matching description time codes and odometry time codes for each stop segment defined by the stop points, taking into account the delay of linguistic descriptions and creating instances from the matched pairs.

The disadvantage of this approach is that it assumes that the delay is the same in each segment which may not be the case. On the other hand, it is the best estimation that can be made. The procedure also allows separating instances into two sets: a clean set where the time codes of descriptions have been shifted by a zero delay and a set of instances where the time codes of descriptions have been shifted by a delay grater than zero. Note that time-shifting was only used for creating motion instances where the timing is crucial but not for the instances of object relations where delays were not experienced.

find_delay: takes $stoptime$ and an ordered list of odometry time codes $OTimes$.

1. Using $stoptime$, $OTimes$ and **find_property_time** find the immediately preceding odometry time $otime_i$.
2. Extract $speed$ from the $otime_i$ entry.
3. If $speed = 0.0$, then $delay = 0$.
4. Else, using $Otimes$, $otime_i$ and $stoptime$.
   (a) $otime_{i-1}$ = preceding($otime_i$,$OTimes$)
   (b) If for $otime_{i-1}$ $speed = 0.0$ then $delay = stoptime - otime_{i-1}$.
   (c) Else recurse (4) using $OTimes$, $otime_{i-1}$ and $stoptime$.
   (d) If $otime_0$ is reached, $delay = error$.
5. Return $delay$.

Figure 3.7: Starting at a description time code $stoptime$, the algorithm regresses to find a preceding odometry entry with a time code $otime$ at which $speed$ is 0. The $delay$ is the difference between the $stoptime$ and $otime$.

3.3.2 Normalising non-numeric data

The configuration of the system and the environment was slightly different while collecting data for the *Simple* and *All* datasets. Because *Simple* is taken as a subset of *All*, this means that its properties relative to the environment and the configuration of the robot would not have a comparable representation in *All* and hence a lot of noise would be added for machine learning. Furthermore, our aim is to use the theories learned in new environments and under new conditions and thus normalisation of data is necessary.

The simplest normalisation steps are required for speed and angular velocity. The system constrains the speed and the heading of the vehicle depending on the value given in the *iAGV/iPlatform* block of the MOOS configuration file. Thus, when creating instances we simply divide every speed and every angular velocity value with the maximum values defined there. Note that because both maximum values are meant as constraints on movement, the vehicle may achieve slightly higher values before these constraints are applied by the system. This means that occasionally we may encounter

normalised values just above 1, particularly in the case of speed.

The normalisation of object coordinates is slightly more involved. We assume that the distances in the room are relative to the size of the room which also defines the maximum distance that can be encountered. The size of the room can be estimated from the SLAM map which contains, as previously shown, coordinates of mapped points relative to the origin of the robot. To find the room size it is not enough to find the extreme value in either positive or negative dimension of the x or y axis on the map. This would only be an estimation of one half of the size of the room on each axis, and when both axes are taken together, only one quarter of its size. There is no guarantee that the origin of the robot is in the centre of the room, and hence multiplying the above values by 2 would also give an incorrect estimation. In fact, the procedure is quite simple. In addition to the maximum value we also need to find the minimum value, thus both extremes, and then calculate the distance between them ($d = abs(Max - Min)$). The d_x and d_y that we obtain define a rectangle that encompasses all the points on the map. Thus, to normalise the coordinates of objects we divide every x coordinate by d_x and every y coordinate by d_y.

3.3.3 Extracting linguistic data

In MOOS logs linguistic data is represented as sentences or their parts. However, for machine learning we require that words are assigned to one of the attributes *Verb*, *Direction*, *Heading*, *Manner* and *Relation*. We also need to identify objects in descriptions so that we can extract their locations for attributes *LO_x*, *LO_y*, *REFO_x* and *REFO_y*. For the *Simple* dataset the task is nearly complete. The describers were restricted to pre-specified categories and lexical entries which can be easily extracted from the structured log files. However, in the *All* dataset the describers could generate any sentence or its part.

We could use an automatic POS tagger to assign words to these categories. However, this was not really necessary. First of all, the number of words used in each

context is relatively small and they are used in a non-ambiguous way. Secondly, many descriptions are incomplete fragments and contain significant errors created by the speech recogniser. Such data would obstruct the tagger. Finally, the categories that we want to tag the descriptions with do not correspond to the typical categories found in POS taggers such as nouns, verbs, adjectives, adverbs and prepositions but are their semantic specialisations which fix the denotation of a particular word to the perception of the environment.

Each such lexical entry is only interpretable in a particular class of situations. For example, in our case we must distinguish two entries for "left", one referring to the direction in which the robot is moving one describing the location of objects. Crangle and Suppes (1994) propose such context-fixing semantic grammars for interpreting natural language commands to robots. The denotations of words are pre-specified procedures similar to semiotic schemas of Roy (2005). For our corpus of descriptions the denotations are learned associations between the attribute values used in each learning task.

To create such lexicon from human descriptions we designed an automated tool called *TagWords*. This ensures that a human tagger only needs to assign a category to each new word form used in a particular setting once. In the first step the tool extracts all words from a given set of log files. Log files containing descriptions of motion and descriptions of object relations are always processed separately. In the extracted set each word is represented once but its representation also contains a count of its occurrence in the log files. The collected words are compared to the words in the lexicon that has been created so far. The lexicon is represented as a set of Prolog predicates (Bratko, 2001) called *lexical_entry/1* as shown in Figure 3.8. The predicate takes one argument which is a list of feature-value pairs. If the collected word, for example "moving", already exists in the lexicon, if there exists a *lexical_entry/1* predicate with a pair *form=moving* on its list, then only the *count* feature on its list is updated with the count

from the log files, for example 32. If the word is not found in the lexicon, the system prompts a human tagger to assign it one of the predefined categories which correspond to our machine learning attributes. Each category imposes a set of lexical feature-value pairs, for example *[cat=v,sem_type=movement, arg_list=[direction,heading,manner]]*. Together with the pairs *form=moving* and *count=32* these are written as a list argument of a new lexical entry predicate. The user also has a chance to assign no category to a given word if the word does not belong to any of the categories that we intend to learn.

If a word has been incorrectly recognised by the speech recogniser, the user can specify its correct form. For example, the second *lexical_entry/1* predicate in Figure 3.8 defines a word "morning" as a verb. The predicate *spelling_correction(morning,v,moving)* defines that the form "morning" is in fact "moving". This mechanism is sometimes used to rewrite a variant of a word to consolidate the class labels for machine learning. As shown in Figure 3.8 "moved" is rewritten as "moving" and "left" as "to_the_left_of".

Once the lexicon is built, it can be used to extract words of the relevant category from log file descriptions such as "Now you're moving forward left". We tokenise sentences as lists of words. Using the lexicon we first find the main predicate from each list of words. For motion descriptions the main predicate is the verb (*cat=v*), for descriptions of object relations this is the relation (*cat=p*). If the main predicate cannot be found, then the sentence is not an interesting description and hence the instance can be discarded. Once "moving" is identified in the sentence above, its lexical entry in Figure 3.8 tells us what arguments we should find together with it on the list of utterance words. This information is encoded in the value of its *arg_list* feature. The names listed here correspond to the semantic types of other words in the lexicon. According to the *arg_list* of "moving" we should find three words whose *sem_types* are *direction*, *heading* and *manner* respectively. We check the list of words for a word with a matching *sem_type* and return the form of this word. Using the lexicon in Figure 3.8 this search extracts "forward" for *direction* and "left" for *heading*. We stop the search for

```
lexical_entry([form=moving,cat=v,sem_type=movement,
   arg_list=[direction,heading,manner],count=811]).

lexical_entry([form=morning,cat=v,sem_type=movement,
   arg_list=[direction,heading,manner],count=5]).

lexical_entry([form=moved,cat=v,sem_type=movement,
   arg_list=[direction,heading,manner],count=2]).

lexical_entry([form=forward,cat=adv,
   sem_type=direction,arg_list=[],count=282]).

lexical_entry([form=left,cat=adv,
   sem_type=heading,arg_list=[],count=739]).

lexical_entry([form=slowly,cat=adv,
   sem_type=manner,arg_list=[],count=495]).

lexical_entry([form=left,cat=p,sem_type=relation,
   arg_list=[object,object],count=65]).

lexical_entry([form=table,cat=n,sem_type=object,
   arg_list=[],count=46]).

lexical_entry([form=chair,cat=n,sem_type=object,
   arg_list=[],count=21]).

spelling_correction(morning,v,moving).
spelling_correction(moved,v,moving).
spelling_correction(left,p,to_the_left_of).
```

Figure 3.8: An extract from the lexicon

a word when the first candidate is found. If no word is matched as in the case of the *manner* argument, we return the word "none" instead. In all cases if the lexical entry is associated with a spelling correction, then the corrected form is returned instead of the original form. In sum, the procedure extracts the following attribute values from the sentence above: *Verb:* "moving", *Direction:* "forward", *Heading:* "left" and *Manner:* "none".

Instead of returning "none" if no word for a category/attribute is found, we could also treat such cases as missing attribute values which Weka algorithms can successfully deal with. However, this would not be appropriate for linguistic descriptions. A non observed linguistic description does not mean that the value is missing but that it has been omitted for the reasons of linguistic relevance. For example the "The robot is moving slowly" does not mean that its heading is unknown but that it is moving in a default direction such as "straight ahead". Thus, there exists a good reason to use a special value such as "none" rather than the missing values. However, this only applies for the adverbial categories of motion descriptions. If the name of one of the objects in a description of object relations is not found, then the description is invalid and the instance should be discarded.

The word extracting procedure only extracts one predicate and its arguments from a string of words and therefore can only deal with mono-clausal descriptions of motion and object relations which represent the majority of descriptions in our corpus. If it encounters a complex description such as "Flakey is directly behind the box and to your right" from Table 3.2, then only the first predicate "behind" is extracted. We manually checked the accuracy of word extraction on a sample of corpus and we concluded that it gives reliable results.

3.3.4 Creating Weka datasets

In the preceding section we discussed how observations were matched and extracted from the MOOS log files. In order for us to be able to use this information with Weka

learners, it must be represented as a set of *instances* as discussed in Section 2.4.3. An instance is an independent example of a *concept* that we want to learn. Each instance is defined by a set of *attributes*, that is some observed properties and measures. All instances in a dataset have to have the same number of attributes and their values must be compatible. A dataset can be represented as a table where the columns define the attributes and the rows the instances.

For Weka the datasets of instances must be in the Attribute-Relation File Format (ARFF). This is a text file with a strictly defined structure. Figure 3.9 shows an example of a learning task from our *Simple* dataset. The file consists of two parts: the header section and the data section. The header defines the attributes and their data types. The data types that can be handled by Weka are *nominal* or categorial data, *numeric* data, *string* data or *date*. These are quite general categories, for example, the statistical literature usually distinguishes numeric data that is ordinal, interval or ratio. The reason for this is that not all classifiers can handle all data types, but if they do, they may not handle them in the same way. Thus, further checks of the consistency of data types are made at the level of the individual classifiers. Also the human preparing the experiment must be confident that a particular data-classifier combination will give sensible results. For attributes that are nominal, we have to declare a list of their possible values. The last attribute declared is interpreted as the concept to be learned, unless specifically configured otherwise in the learning process. The data section contains a comma separated list of attribute values for each instance. The consistency of a file is checked when it is loaded into one of the Weka programmes.

Although Weka contains a number of tools that assist the conversion and filtering of data into the ARFF (see Witten and Frank, 2005, page 380ff.), we have chosen to implement our own method of creating such files from the instance files produced by our instance creators. This method allows us to create various configurations of attributes and subsets of instances on the fly and almost entirely without human intervention.

```
@relation object_relation

@attribute lo_x numeric
@attribute lo_y numeric
@attribute refo_x numeric
@attribute refo_y numeric
@attribute relation {behind, in_front_of, \
   to_the_right_of, to_the_left_of}

@data
-0.2223, -0.0115, 0, 0, to_the_left_of
0.1834, 0.0005, 0, 0, to_the_right_of
-0.1037, 0.1157, 0, 0, to_the_left_of
0.0862, 0.2081, 0, 0, in_front_of
0.1834, 0.0005, 0, 0, to_the_right_of
...
```

Figure 3.9: An excerpt from an ARFF file from the *Simple* dataset to learn the meanings of object relations

This way, the entire learning procedure can be made automatic. Our instance creators create instances as Prolog predicates and hence it is quite straightforward to implement various selection and search mechanisms and subsequently write the data as comma separated text files.

The instances are prepared for Weka with *Arff writer*. This works in three stages. In the first stage a set of instance files is loaded and the values of nominal attributes in these files are automatically collected. These are later declared in the header of the arff file. During the second stage, user input is required. The user is presented with a list of possible attributes from which they can select the ones that they want to write to a file. By convention, the last attribute must be the target concept to be learned. The user is also queried whether they want to write out all the instances or only those for which the time-shifting algorithm found a zero delay. We call the resulting datasets *Time-shifted* and *Zero-Time-shifted*. Of course, this only applies to instances that were created with time-shifting. If no time-shifting is applied during dataset creation, we call such a dataset *Not-Time-shifted*. Finally, the user has to choose the name of the

arff file to which the data will be written. In the third stage, *Arff writer* writes out the file. Following the recommendations of the Weka authors, we include human readable comments preceding the header: we record the source instance files from which the dataset was produced, the date and time of the generation, and a statement whether the data is *Time-shifted*, *Zero-Time-shifted* or *Not-Time-shifted*. This information is intended to help the identification of a dataset. The header and the data are then automatically formatted and written out, but only for the attributes and the instance subset selected by the user.

The learning algorithms that we chose for the learning tasks (see Section 3.4) are classification algorithms and cannot handle numeric prediction. For example, they predict a nominal class rather than a rational number. However, we would also like to build classifiers that would predict a number, for example, what speed should the robot achieve if a user requests "go forward slowly". Although there are learners that can handle numeric prediction, for example the linear regression learner, we chose not to use them, mostly to ensure a comparison between all our learning tasks. Instead, we *discretised* the numeric continuum into discrete intervals and matched each rational number with one of them. We thus ended up with nominal attributes which could be used with the classification learners. The discretisation step was added as an option to *Arff writer*.

But how many nominal classes should be created? Without a doubt, a considerable amount of information is lost in discretisation since exact numeric values are transformed into generalised nominal categories. Thus, classifiers predicting nominal classes rather than discrete numeric values will be less accurate in referring to the state of the real world. In this respect, it is desirable to create a large number of nominal classes which approximate better the underlying numeric continuum. On the other hand, we expect that the relation of the numeric attribute to other nominal attributes is not random. This means, that if the numeric attribute is discretised to a large number

of classes, the relation will be blurred. Suppose that we created two classes from one natural class which we do not know. The classifier would create two separate classification paths where in a perfect situation one would suffice and hence the learned theory will be unnecessarily more complex and would appear less readable to humans, if this is one of the priorities of our learning. The increased number of classes also creates sparse data. With a large number of classes, only a few instances would fall under each nominal class and the learner would be less successful in making generalisations about them. From the perspective of the learner a small number of nominal classes is preferred. Ideally, this would just be a single nominal class which would make classification extremely straightforward as the classifier would assign the same class to every instance and be accurate 100% of the time. However, such classification would be useless. Thus, considerable care is needed when choosing the number of classes when discretising a numeric attribute to balance between the richness of the predictability and the performance of the trained classifiers.

How the intervals span across the numeric continuum is also important. For example, if the minimum and the maximum value are identical but their signs are reversed, then creating an even number of intervals will leave us with 0 as the interval boundary. This may not be desirable because we expect that certain attribute values, for example the description "stopped", will show a relation with the values of the numeric attribute clustering around 0, for example Speed $= 0$. If we discretise the numeric attribute this way, the zero category is lost. Therefore, if both the minimum and the maximum value are identical but with reversed signs, an odd number of intervals is preferred. This creates intervals of equal width and also ensures that one of them spans across the zero value with equal interval boundaries in the positive and in the negative dimension.

To find the extreme value of a numeric attribute, its minimum and maximum, we implemented two methods. In most cases the extreme value is known to the user, especially since numeric values are normalised during instance creation and thus their

values are either -1 or 1. The user can enter such values manually. The system assumes that the negative and the positive extremes are identical and hence only one number is required. However, taking absolute extreme values for the lower and upper limits of intervals may not be an ideal choice. Experience shows that in certain cases the sampled values are not equally spread through the range and may never approach such extremes. The majority of values usually lie in the bottom quarter of the range. Thus, when using a small number of intervals it may happen that all the values are assigned to one or two categories with a consequence that significant attribute information is lost. To counter this difficulty we implemented another method where the discretisation procedure determined the extreme value from the current values of each attribute. The extreme value is either the negative or the positive extreme found, whichever is greater in terms of its absolute value. As before, this value is taken as both the minimum and the maximum for creating intervals. It is this method that was used for discretising numeric attributes for datasets in our learning experiments.

The class labels of discretised numeric attributes preserve the values of interval boundaries in their names. Thus, after we use classifiers to predict one of these nominal classes, we can extract the interval boundaries from its value and generate a real number in that range. We return to this issue when describing the properties of *pDialogue* in Chapter 4.

Table 3.4 lists all attributes and their values for both *Simple* and *All* datasets for each learning task created with *Arff writer*. The last attribute in each section of the table is the target concept that is learned. In this particular case all numeric attributes have been discretised to 7 bins when used as target concepts.

Attributes	Values	
Verb		
Simple	Delta-Heading	*numeric*
	Speed	*numeric*
	Verb	moving, stopped
All	Delta-Heading	*numeric*
	Speed	*numeric*
	Verb	going, edging, continuing, reversing, creeping,

Attributes	Values	
		turning, moving, stopped
Direction		
Simple	Delta-Heading	*numeric*
	Speed	*numeric*
	Direction	backward, forward, none
All	Delta-Heading	*numeric*
	Speed	*numeric*
	Direction	stopped, spot, backward, forward, none
Heading		
Simple	Delta-Heading	*numeric*
	Speed	*numeric*
	Heading	right, left, none
All	Delta-Heading	*numeric*
	Speed	*numeric*
	Heading	anticlockwise, clockwise, straight_ahead, hard, straight_line, around, straight, 180, right, left, none
Manner		
Simple	Delta-Heading	*numeric*
	Speed	*numeric*
	Manner	fast, moderately, slowly, none
All	Delta-Heading	*numeric*
	Speed	*numeric*
	Manner	quickly, walking_pace, imperceptible, tightly, gently, rapidly, fast, moderately, slowly, none
Relation		
Simple	LO_x	*numeric*
	LO_y	*numeric*
	REFO_x	*numeric*
	REFO_y	*numeric*
	Relation	behind, in_front_of, to_the_right_of, to_the_left_of
All	LO_x	*numeric*
	LO_y	*numeric*
	REFO_x	*numeric*
	REFO_y	*numeric*
	Relation	next, after, near, parallel, opposite, facing, front, far, right, left, behind, close
Delta-Heading, 7 bins		
Simple	Verb	moving, stopped
	Direction	backward, forward, none
	Heading	right, left, none
	Manner	fast, moderately, slowly, none
	Delta-Heading	-0.6001…-0.4286, -0.4286…-0.2572, -0.2572…-0.0857, -0.0857…0.0857, 0.0857…0.2572, 0.2572…0.4286, 0.4286…0.6001
All	Verb	edging, continuing, creeping, reversing, going, turning, moving, stopped
	Direction	spot, backward, forward, none
	Heading	straight_ahead, hard, straight_line, around, 180, anticlockwise, straight, clockwise, right, left, none
	Manner	quickly, walking_pace, imperceptible, tightly, gently, rapidly, fast, moderately, slowly, none
	Delta-Heading	-0.8791…-0.6279, -0.6279…-0.3768, -0.3768…-0.1256, -0.1256…0.1256, 0.1256…0.3768, 0.3768…0.6279, 0.6279…0.8791
Speed		
Simple	Verb	moving, stopped
	Direction	backward, forward, none
	Heading	right, left, none

Attributes	Values	
	Manner	fast, moderately, slowly, none
	Speed	-1.0055…-0.7182, -0.7182…-0.4309, -0.4309…-0.1436, -0.1436…0.1436, 0.1436…0.4309, 0.4309…0.7182, 0.7182…1.0055
All	Verb	edging, continuing, creeping, reversing, going, turning, moving, stopped
	Direction	spot, backward, forward, none
	Heading	straight_ahead, hard, straight_line, around, 180, anticlockwise, straight, clockwise, right, left, none
	Manner	quickly, walking_pace, imperceptible, tightly, gently, rapidly, fast, moderately, slowly, none
	Speed	-1.0742…-0.7673, -0.7673…-0.4604, -0.4604…-0.1535, -0.1535…0.1535, 0.1535…0.4604, 0.4604…0.7673, 0.7673…1.0742
LO_x, 7 bins		
Simple	Relation	behind, in_front_of, to_the_right_of, to_the_left_of
	REFO_x	*numeric*
	REFO_y	*numeric*
	LO_x	-1.0…-0.7143, -0.7143…-0.4286, -0.4286…-0.1429, -0.1429…0.1429, 0.1429…0.4286, 0.4286…0.7143, 0.7143…1.0
All	Relation	next_to, after, near, parallel_to, opposite_of, facing, far_from, close_to, behind, in_front_of, to_the_right_of, to_the_left_of
	REFO_x	*numeric*
	REFO_y	*numeric*
	LO_x	-1.0…-0.7143, -0.7143…-0.4286, -0.4286…-0.1429, -0.1429…0.1429, 0.1429…0.4286, 0.4286…0.7143, 0.7143…1.0
LO_y, 7 bins		
Simple	Relation	behind, in_front_of, to_the_right_of, to_the_left_of
	REFO_x *numeric*	
	REFO_y *numeric*	
	LO_y	-1.0…-0.7143, -0.7143…-0.4286, -0.4286…-0.1429, -0.1429…0.1429, 0.1429…0.4286, 0.4286…0.7143, 0.7143…1.0
All	Relation	next_to, after, near, parallel_to, opposite_of, facing, far_from, close_to, behind, in_front_of, to_the_right_of, to_the_left_of
	REFO_x	*numeric*
	REFO_y	*numeric*
	LO_y	-1.0…-0.7143, -0.7143…-0.4286, -0.4286…-0.1429, -0.1429…0.1429, 0.1429…0.4286, 0.4286… 0.7143, 0.7143…1.0
REFO_x, 7 bins		
Simple	Relation	behind, in_front_of, to_the_right_of, to_the_left_of
	LO_x	*numeric*
	LO_y	*numeric*
	REFO_x	-1.0…-0.7143, -0.7143…-0.4286, -0.4286…-0.1429, -0.1429…0.1429, 0.1429…0.4286, 0.4286…0.7143, 0.7143…1.0
All	Relation	next_to, after, near, parallel_to, opposite_of, facing, far_from, close_to, behind, in_front_of, to_the_right_of, to_the_left_of
	LO_x	*numeric*
	LO_y	*numeric*
	REFO_x	-1.0…-0.7143, -0.7143…-0.4286, -0.4286…-0.1429, -0.1429…0.1429, 0.1429…0.4286, 0.4286…0.7143,

Attributes	Values	
		0.7143...1.0
REFO_y, 7 bins		
Simple	Relation	behind, in_front_of, to_the_right_of, to_the_left_of
	LO_x	*numeric*
	LO_y	*numeric*
	REFO_y	-1.0...-0.7143, -0.7143...-0.4286, -0.4286...-0.1429, -0.1429...0.1429, 0.1429...0.4286, 0.4286...0.7143, 0.7143...1.0
All	Relation	next_to, after, near, parallel_to, opposite_of, facing, far_from, close_to, behind, in_front_of, to_the_right_of, to_the_left_of
	LO_x	*numeric*
	LO_y	*numeric*
	REFO_y	-1.0...-0.7143, -0.7143...-0.4286, -0.4286...-0.1429, -0.1429...0.1429, 0.1429...0.4286, 0.4286...0.7143, 0.7143...1.0

Table 3.4: Attributes and their values for different learning tasks

3.4 The learning algorithms

We chose two popular algorithms from the Weka toolkit to train on our data: *Naive-Bayes* and *J48*. The latter is the Weka's implementation of the ID3/C4.5 decision tree learner developed by Quinlan (1986, 1993). Each algorithm forms hypotheses about the data quite differently. Naive-Bayes is a statistical learner based on the Bayesian rule of conditional probability (Bayes, 1763), whereas the decision tree learner is primarily a symbolic learning technique. The choice thus reflects a well-known opposition between statistical and symbolic techniques in computational linguistics in general. The learners vary significantly in their complexity and the complexity of structures that they build. *Naive-Bayes* is a very simple approach, even simpler than the most basic variant of the decision tree learner known as ID3. This has been further improved by various optimisation techniques and is consequently known as C4.5. Experience shows that *Naive-Bayes* can often compete in performance with other more complex systems.

A comparison between the learners is useful because different machine learners use different methods of generalisations and output knowledge representations which may not be suitable for every dataset. For example, a technique such as a decision tree

learner which assigns a particular target class to every instance based on the values of other attributes may be less successful if the dataset contains regularities only between subsets of attributes. Consequently, certain regularities may be missed, or in the best case, the output classification structure may be very complex and non-intuitive.

In the following two subsections we give an overview of both approaches. We base our discussion on two standard textbooks. Naive-Bayes classifier is described in (Mitchell, 1997, Chapter 6) and (Witten and Frank, 2005, Section 4.2). The ID3/C4.5/J48 algorithm is described in (Mitchell, 1997, Chapter 3) and (Witten and Frank, 2005, Section 4.3 and Section 6.1).

3.4.1 Naive-Bayes learner

Naive-Bayes learner models probabilistic knowledge. It assumes that the attribute data is governed by probability distributions and that learning involves estimating these distributions from the learning dataset. Once the distributions are determined, they can be used on new data to predict the best or the most probable classification.

Let us first consider how the Bayes theorem can be used for machine learning in general. If h is the hypothesis that we want to learn and E is the evidence on which the hypothesis is based, then the relation between the two can be expressed as a conditional distribution $P(h|E)$. According to the Bayes theorem this can be calculated from the following equation:[4]

$$P(h|E) = \frac{P(E|h)P(h)}{P(E)} \tag{1}$$

$P(h|E)$ is also called *posterior* probability of h because it expresses the probability of h after evidence E is examined. On the other hand, $P(h)$ and $P(E)$ are known as *prior* probabilities. For machine learning $P(h)$ is particularly useful, because it allows integrating background knowledge for that hypothesis. Background knowledge is the

[4]The notation $P(A)$ denotes the probability of the event A and $P(A|B)$ denotes the probability of event A given event B.

knowledge that we may have about the validity of the hypothesis before the evidence is examined. It follows from Equation 1 that the posterior probability $P(h|E)$ increases while increasing $P(E|h)$ and $P(h)$ and decreases while increasing $P(E)$. The latter is because the more likely we are to find the evidence E independently of h, the less influence has E on h.

In a classification learning task we represent each value of the target concept as a separate hypothesis h. Together they form a set of hypotheses H. The task of classification is to find the most probable hypothesis h in this set using the data from the instance that we want to classify. We calculate the posterior probabilities for every hypothesis in H and choose the one with the maximum value. This hypothesis is also known as a *maximum a posteriori hypothesis* (MAP).

$$h_{MAP} \equiv \arg\max_{h \in H} P(h|E) \tag{2}$$

One of the benefits of a statistical method is that the classification not only determines a class of an instance but also gives the probability with which this instance belongs to that class. However, if we are only interested in the class, Equation 1 can be further simplified by excluding $P(E)$ which is constant for each h in H. Furthermore, if all hypotheses have the same a priori probability, then $P(h)$ can be excluded too giving us the form

$$h_{ML} \equiv \arg\max_{h \in H} P(E|h) \tag{3}$$

$P(E|h)$ is the probability/likelihood of E being produced by the hypothesis h. For this reason h_{ML} is often called a *maximum likelihood hypothesis*.

The Bayes theorem can be applied to machine learning tasks in many different ways as described in (Mitchell, 1997, Chapter 6). Here we are interested in a method known as Naive-Bayes. According to this method we apply the Bayes theorem directly to the values of the instance attributes to predict our hypothesis or the target class. The

evidence E in Equation 1 and 2 is in this case simply an ordered list of attribute values $\langle a_1, a_2 ... a_n \rangle$. The hypothesis h is the value of the target class that we want to predict. Both equations can be rewritten as follows:

$$v_{MAP} \equiv \arg\max_{v_j \in V} P(v_j | a_1, a_2 ... a_n) \tag{4}$$

$$v_{MAP} \equiv \arg\max_{v_j \in V} \frac{P(a_1, a_2 ... a_n) P(v_j)}{P(a_1, a_2 ... a_n)} \tag{5}$$

$$\equiv \arg\max_{v_j \in V} P(a_1, a_2 ... a_n | v_j) P(v_j) \tag{6}$$

We estimate $P(v_j)$ by finding the percentage frequency with which the value v_j occurs in the training set. To estimate $P(a_1, a_2 ... a_n | v_j)$ in the same way we would have to count how many times a sequence of attribute values $a_1, a_2 ... a_n$ occurs with a particular target value v. The number of terms to count equals the number of combinations of a sequence $a_1, a_2 ... a_n$ ($n!$) times the number of values of v. In practice this is almost never possible because our dataset is never big enough to observe each term a number of times. Instead, we naïvely assume that the attributes $a_1, a_2 ... a_n$ are conditionally independent in which case the probability $P(a_1, a_2 ... a_n | v_j)$ equals to $\prod_i P(a_i | v_j)$. In this case the number of terms to count equals the total number of values of all attributes times the number of values of v. Hence, a Naive-Bayes classifier is defined as follows:

$$v_{NB} = \arg\max_{v \in V} P(v_j) \prod_i P(a_i | v_j) \tag{7}$$

Let us consider a simple example. Assume that we have a training dataset of 20 instances that are defined by three attributes: a linguistic category *Verb* which can take the values "moving", "reversing" and "stopped", another linguistic category *Manner* with the values "none", "slowly" and "fast", and a category *Speed* representing the speed of the robot. This numeric attribute has been discretised to nominal categories S1, S2 and S3 and represents the target class that we want to learn. The learning step includes establishing counts and the associated probabilities represented in Table 3.5.

The table cross-tabulates the value of each attribute with each target class value to give us the estimates of $P(a_i|v_j)$. The numbers in brackets contain the actual counts. The second row of the table contains the counts and the associated probabilities of the times each target class value occurs in the dataset regardless of the attribute value. These are our prior probabilities of the target class values $P(v_j)$.

Attribute	Value	S1	S2	S3
		0.3 (6/20)	0.2 (4/20)	0.5 (10/20)
Verb	moving	0.5 (3/6)	0.25 (1/4)	0.8 (8/10)
	reversing	0.333 (2/6)	0.25 (1/4)	0.1 (1/10)
	stopped	0.167 (1/6)	0.5 (2/4)	0.1 (1/10)
Manner	none	0.333 (2/6)	0.5 (2/4)	0.3 (3/10)
	slowly	0.5 (3/6)	0.25 (1/4)	0.5 (5/10)
	fast	0.167 (1/6)	0.25 (1/4)	0.2 (2/10)

Table 3.5: An example Bayesian classifier

What is the predicted value of *Speed* if the linguistic description is "moving fast"? To find the most likely value S we need to read off the estimated probabilities from Table 3.5 and apply them to Equation 7. The predicted target class value S is the one that maximises $P(S) = P(S) \times P(moving|S) \times P(fast|S)$. As shown in Equations 8–10, this target class is $S3$.

$$P(S1) = 0.3 \times 0.5 \times 0.167 = 0.025 \quad (8)$$

$$P(S2) = 0.2 \times 0.25 \times 0.25 = 0.013 \quad (9)$$

$$P(S3) = 0.5 \times 0.8 \times 0.2 = 0.08 \quad (10)$$

The assumption about the independence of attributes has a negative effect on the performance of classifiers. If the values of two attributes correspond completely to one another, a certain value of the first attribute a_1 will always co-occur with a certain value of the second attribute a_2. If this is so, one of the attributes is redundant, as the target value v can be predicted equally well just by knowing either the value of a_1 or

a_2. Thus, for the purpose of learning the target value, the attribute pair $\langle a_1, a_2 \rangle$ functions as a single attribute. The probability $P(a_1|v_j)$ is equal to the probability $P(a_2|v_j)$ and when both probabilities are multiplied in Equation 7, the weight of the attribute pair is squared. The attribute pair wrongly receives more weight than other truly independent attributes contributing to the prediction of the target value and the learning is skewed toward that pair. The problem can be overcome by carefully selecting the attributes for the learning process and ensuring that they are independent.

We may encounter another difficulty using Equation 7 when a certain attribute value does not occur with a target value at all. This is because we estimate $P(a|v)$ with the fraction $\frac{n_{v,a}}{n_v}$ where n_v is the number of instances where the target value is v and $n_{v,a}$ is the number of instances where the target value is v and the value of the attribute in question is a. If the latter is 0, it follows that $P(a|v)$ is also 0. When this is applied to Equation 7, the overall probability defined by the product turns out 0 too, regardless of the other probabilities in the product. The difficulty can be resolved by assuming that each $P(a|v)$ has some small a priori probability and so an attribute value whose count is 0 does not receive a zero probability. This can be done by changing the fraction as follows:

$$P(a_k|v) = \frac{n_{v,a_k} + mp}{n_v + m} \tag{11}$$

a_k indicates a member of the set of values of attribute a, K. m is an arbitrary constant. The same value of m must be applied when calculating $P(a_k|v)$ for all members $a_{k \in K}$. The values of p must sum up to 1 for all members $a_{k \in K}$. If we assume that the prior probabilities are equal for all $a_{k \in K}$, p must be equal to $\frac{1}{k}$. Once enough data is considered, the priors are adjusted accordingly to reflect the true probabilities of values of $a_{k \in K}$ in the dataset. The size of m affects the importance of the priors: the larger its value, the more weight the priors receive. The intuition behind Equation 11 is that m are virtual samples of the values $a_{k \in K}$ that are distributed according to p. The more

virtual samples with equal distribution of values we add, the greater is the effect of these samples on the true distributions reflected in the data. However, with a reasonable amount of new data, the virtual samples will have very little effect and the method performs well in practice. Frequently, k is taken for the value of m which ensures that the initial count in the nominator is 1 rather than 0 ($mp = k \times \frac{1}{k}$). This method is known as the *Laplace estimator* (Witten and Frank, 2005, pages 91–92).

Our data also includes attributes that are numeric and cannot be used with Equation 11 to determine the probabilities $P(a|v)$. One solution is to discretise the numeric interval to nominal classes as discussed before. Another, better solution is to model the attribute values by some statistical distribution and use standard statistical estimation techniques for that distribution. Most frequently it is assumed that the values of a numeric attribute a that occur with each target class v are samples of an underlying normal distribution. If so, we can determine their sample mean μ and their sample standard deviation σ which concludes the learning step. For classification, to calculate the probability that a particular value x of a numeric attribute a predicts the target class v ($P(a_x|v)$), we apply the probability density function for normal distribution which is as follows (see Witten and Frank, 2005, page 93):

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{12}$$

This function returns the probability of a particular value x being drawn from the underlying normal distribution with a sample mean μ and standard deviation σ.

Finally, let us consider how the Bayesian approach deals with missing attribute values. If an instance contains an attribute a_i whose value is missing during classification, then $P(a_i|v_j)$ for a_i is simply omitted from the product in Equation 7. Since it is omitted from calculating the probabilities for all $v \in V$, the attribute a_i makes no influence on predicting the class v. If an attribute value is missing during training, then it is simply not included in the count in Equation 11. Adding more observations to training can

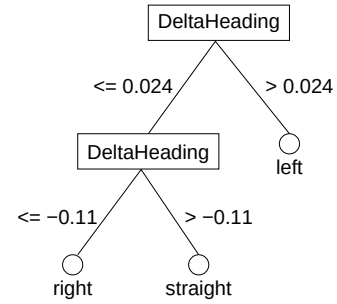
incrementally increase and decrease the probability of v_j. The statistical approach is thus very flexible in handling data.

3.4.2 Decision tree learner (J48)

J48 is a Weka's implementation of the ID3 algorithm and its successors C4, C4.5 and See5/C5.0 developed by Quinlan (1986). A decision tree is a data structure in which each branch node represents a choice or a test and each leaf represents a decision. In decision trees created by machine learning, branches correspond to attributes and leaves to their values. To classify an instance we check the values of its attributes against the attributes in the tree by following the path from the root node to the final terminal node. The value of the terminal node is the value of the target class. A decision tree can also be represented as a set of if-then rules or a set of conjunctions and disjunctions of constraints on the attribute values of instances. The attributes in a path from the root node to the terminal leaf form a conjunction of constrains, whereas the attributes of the neighbouring branches at the same level represent a disjunction of constraints. Figure 3.10 shows a simple decision tree to predict a linguistic description *Heading* ("straight", "left" or "right") that we induced from one of our datasets also rewritten as a set of constraints. If a robot is moving with *Speed* 0.6 and *Delta-Heading* -0.2 then the predicted description of heading is "right" by following the leftmost path in the tree.

A decision tree is learned by taking an attribute and creating a leaf for each of its values. This splits instances into subsets which have an identical value of that attribute. The procedure is repeated recursively on each subset of instances by selecting another attribute. A tree is built when all the training instances at the leaf have the same target value,[5] or when all attributes have been used in that particular path of a tree. Since the algorithm does not use any backtracking it is an example of a greedy search which only explores a single search path for the best candidate tree.

[5]With noisy data this may not be possible and hence this criterion must be relaxed.



$$
\begin{array}{ll}
 & (\text{Delta-Heading} \leq .024 \wedge \text{Delta-Heading} \leq -.11 \wedge \text{Heading = right}) \\
\vee & (\text{Delta-Heading} \leq .024 \wedge \text{Delta-Heading} > -.11 \wedge \text{Heading = straight}) \\
\vee & (\text{Delta-Heading} > .024 \wedge \text{Heading = left})
\end{array}
$$

Figure 3.10: A decision tree that predicts a description for *Heading* also rewritten as rules

The choice of attributes for the root node and all subsequent branches is crucial for producing good classification trees. The trees should separate training instances in such a way that in an ideal case all instances that reach terminal nodes have the same target class. We also prefer trees that are small, with the shortest path from the root node to the terminal leaves. Both requirements are satisfied by a statistical property called *information gain* which is used by ID3 to create each branch of a tree.

The purity/impurity or homogeneity of a set of instances in respect to their target value is measured by *information value* or *entropy* which was adopted by the ID3 algorithm from information theory. If S is a set of instances which have binary target values ($\oplus\ominus$), then the entropy of S is defined as

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \tag{13}$$

$p_{\oplus}$ and $p_{\ominus}$ are the proportions of positive and negative examples in S – for example 11/16 and 5/16. The reason why the terms in Equation 13 are prefixed by a minus is that the logarithms of fractions are negative and hence this ensures an overall positive value of entropy. The value of the entropy function ranges from 0 to 1. If all members

of S are either positive or negative ($p_{\oplus} = 1$, $p_{\ominus} = 0$ or $p_{\oplus} = 0, p_{\ominus} = 1$), the entropy is 0.[6] If the number of positive and negative examples is equal ($p_{\oplus} = p_{\ominus} = 0.5$), the entropy is 1. If the number of positive and negative examples is not equal, the entropy is between 0 and 1. The entropy approaches 0 when the proportion of one of the values increases from 0.5 to 1.

Entropy is used in information theory to measure the amount of information in a transmitted message or how many bits on average is required in a transmission to encode the value of a random member of S.[7] If $p_{\oplus}$ or $p_{\ominus} = 1$, the value of the transmitted member of S is known in advance and thus no bits are required to encode the message. If $p_{\oplus} = p_{\ominus} = 0.5$, the value of each transmitted member has to be specified separately and thus each will take up 1 bit. If either $\oplus$ or $\ominus$ is greater than 0.5, the occurrence of its values is more likely and hence a group of messages can be encoded together. This means that on average less than 1 bit is required to transmit each individual message.

Entropy can be also calculated for a set of members whose values are not binary. In this case Equation 13 becomes

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i \tag{14}$$

where c is the number of classes and p_i is the proportion of members of S belonging to each class c. As before $p_1, p_2 ... p_n$ must sum to 1.

As said before, when selecting an attribute to create a branch of a decision tree, we prefer the attribute that creates the purest or the most homogeneous classes in respect to the target value of instances. The homogeneity can be determined by evaluating the difference in entropy of target class values of a set of instances before the attribute is applied to create a branch and the entropy of subsets that are created after the attribute is applied. The attribute that provides the highest reduction in entropy is the preferred candidate for creating a branch. The reduction of entropy is also called *information*

[6] The approach assumes that $0 \log 0 = 0$ rather than being undefined.
[7] This is why logarithm base 2 is used in Equation 13.

gain. The reason for this naming is that it tells us how much information was *gained* in encoding the value of the target class given that we know the value of one particular attribute. It can be formalised in the following equation:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{15}$$

S is the set of instances before the attribute is applied and S_v are sets that are created by partitioning S by attribute A according to its value v. The average entropy of these sets can be determined by weighting each $Entropy(S_v)$ by the proportion of instances of S that belong to each S_v and summing all the weighted entropies.

One of the problems with *information gain* is that it is biased toward highly branching attributes. These create smaller sets which are more likely to be homogeneous. Consider the extreme case where each instance has a unique value of some attribute such as ID or date. The entropy of each subset containing exactly one member determined with Equation 13 will be 0. Such attribute perfectly predicts the target class. Thus, when information gain is calculated with Equation 15, this will simply be $Gain(S, A) \equiv Entropy(S)$, a maximally attainable value. The attribute would be the preferred attribute to create a branch on but the tree would not be a very informative one. It would be completely flat.

Various improvements have been suggested to replace the notion of *information gain* (for an overview see Mitchell, 1997, page 74). Quinlan (1986) introduces the *gain ratio*. This measure incorporates a penalty for increased number of splits based on the attribute values and a penalty for uniform assignment of instances into the split subsets. Both properties can be captured by entropy. The approach simply calculates the entropy of a set S in respect to the values of the attribute under consideration ($v \in V(A)$) using Equation 14 which we rewrite as Equation 16. $\frac{|S_i|}{|S|}$ is the proportion of S having a particular value v of attribute A. Note that this calculation is unrelated to the

calculation of entropy for the purposes of information gain. There we evaluated the entropy of a set in respect to the value of the target class.

$$Entropy(S, A) \equiv \sum_{i=1}^{v} -\frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|} \tag{16}$$

The *gain ratio* is defined as the relationship between the information gain and the information value of the attribute:

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{Entropy(S, A)} \tag{17}$$

Given a constant information gain $Gain(S, A)$, the greater the entropy $Entropy(S, A)$, the smaller the $GainRatio(S, A)$. The attribute with the highest gain ratio is the preferred one. The gain ratio becomes problematic in cases where all or almost all members of S have an identical value of the attribute A in which case $|S_i| \approx |S|$. In these cases the value obtained by Equation 17 becomes undefined or very large when $Entropy(S, A) = log_2 1 = 0$ or approximately 0. The latter results in preferring an attribute just because its intrinsic information value is very low. The solution proposed by Quinlan (1986) is to choose an attribute with the highest gain ratio but only if its information gain is greater than or equal to the average information gain of all attributes considered.

The decision tree in Figure 3.10 contains splits on the *Delta-Heading* attribute which is numeric but we have not yet discussed how such splits are made. We want to create binary splits. Creating a split discretises a continuous numeric attribute at some threshold t into two categories $A \leq t$ and $A > t$. A useful threshold to split on is such that creates discrete categories with the highest information gain.

Delta-Heading	-1	0	1	2	3	4
Heading	right	straight	straight	left	left	left

Table 3.6: Where to split the *Delta-Heading* attribute?

Table 3.6 shows simplified data which gives rise to a decision tree similar to the one in Figure 3.10. We first sort instances according to the ascending value of the attribute we want to split (*Delta-Heading*). Since we do not want to split items of the same target class, we are only left with two possible positions for the split which correspond to the points where the target class changes. To determine the threshold t the mean of the neighbouring values is taken. This gives us $t_1 = (-1 + 0)/2 = -0.5$ and $t_2 = (1 + 2)/2 = 1.5$. Then, entropy is calculated for both scenarios and the split that minimises its value is the preferred one.[8] In our case this is t_2 as shown below. The square brackets contain the counts of each target class value ("right", "straight" and "left") in each partition created.

$$t_1 : Ent([1,0,0],[0,2,3]) = 1/6 \times Ent([1,0,0]) + 5/6 \times Ent([0,2,3]) = 0.809 \text{ bits}$$

$$t_2 : Ent([1,2,0],[0,0,3]) = 3/6 \times Ent([1,2,0]) + 3/6 \times Ent([0,0,3]) = 0.459 \text{ bits}$$

The discretised numeric attribute subsequently competes for selection in the same way as any other nominal attribute. Nominal attributes can only be used once to create a branch in each path of a tree. Numeric attributes, on the other hand, can be used again to create another split, provided that the split maximises information gain as exemplified in Figure 3.10. This may create trees that are difficult to read, especially if the splits on the numeric attribute do not occur together.

There are a few solutions to how to treat instances with missing attribute values when building decision trees. One of them is to consider missing values as a special value of that attribute. In our case the missing values are processed using this method already before the data reaches the learner. Another possibility is to replace the missing attribute value in a particular instance with the value of that attribute that is most common among instances at the branch we are considering. Even better, we can assign

[8] We do not need to calculate information gain in this case. The entropy of the set before either of the two split scenarios is applied is constant and hence the scenario with the lowest information value will also contribute to the highest information gain.

it the value of that attribute that is encountered with instances that have the majority target class at that branch. In C4.5 (Quinlan, 1993) a more sophisticated method is implemented. First, for each value of the selected splitting attribute we estimate its proportion/probability in the set of instances that have reached that branch. Then we split the instance with the missing value according to these probabilities along each leaf of the branch. The information gain is calculated as usual, but with fractions of instance counts rather than complete counts. Missing values have no effect on the calculation of information gain of their own attribute. However, the instance with a missing attribute value is distributed in proportions along the leaves created by the attribute branch and so it is allowed to make a contribution to the selection of other attributes for subsequent branches but for which it is not missing a value. The same procedure can be used to classify an instance with a missing attribute value using an existing decision tree. In this case the target class is the most probable class in respect to the weights at the leaf node.

A central issue when building any classifier is to ensure that it not only perfectly classifies the training data but also that it performs well on new data that it has not encountered before. The difference in performance is particularly striking if the training data contains noise in the form of incorrect classifications or when the training set of instances is too small and thus unrepresentative of the true relations between the attributes. In this case it is said that the classifier *over-fits* the training examples. Assuming that there are relations between the values of the attributes that can be represented as a tree, errors in the training data will increase the number of branches in a tree thus making trees unnecessarily complex. More splits have to be introduced to accommodate the variations. It can be demonstrated that although building more complex trees increases the accuracy of the classifier on the training data, introducing new branches after some threshold has a negative effect when the classifier is applied on the new data (see Mitchell, 1997, page 67, Figure 3.6).

The optimisation of decision trees is known as *pruning*. Pruning can be done either at the stage when the tree is built (*pre-pruning* or *forward pruning*) or after it has been built (*post-pruning* or *backward pruning*). The latter proves to be more successful in practice. This is because it is more difficult to decide when to stop building a tree rather than build a tree and optimise it later. It may also occur that certain attributes make informative classifications only in combination with other attributes which would not be discovered if the tree would be pruned at that point.

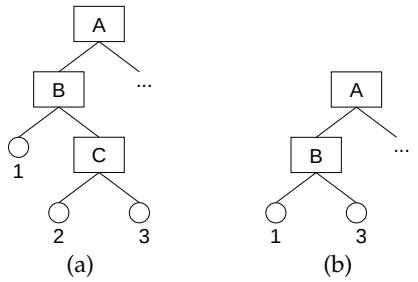


Figure 3.11: *Subtree replacement:* the subtree C is replaced with a leaf 3. This leaf now combines instances that were previously classified under leaves 2 and 3. The class 3 is chosen for this leaf because this is the most common class among the instances that reach it.

Post-pruning can be done using one of the two methods. According to the first method we start at the bottom of a tree, check each decision node and *replace the subtree* defined by that node with a single leaf as shown in Figure 3.11. The class of the leaf is determined by examining the training examples that reach that leaf and choosing the most frequent class. According to the second method *a subtree is raised* to replace another subtree that was removed as demonstrated in Figure 3.12. The instances that were classified under different paths of the subtree that is removed must be reclassified to the paths of the raised attribute.

Pruning is controlled by comparing the error rate that both trees make. The pruned tree is preferred if it performs as well as or better than the original tree. We do not need to evaluate the error rate of the entire tree but we only look at the local error rates created by the nodes or leaves before and after one of the above pruning techniques

is applied. This is because each path in a tree is independent of other paths. Nodes that contribute to the greatest increase in performance are pruned first and the process stops when the performance of the tree cannot be further improved.

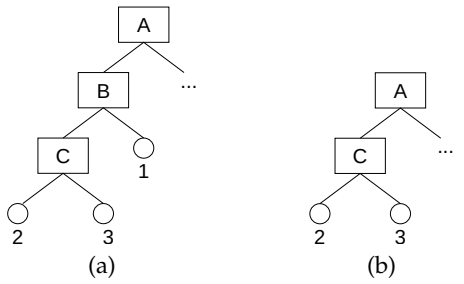


Figure 3.12: *Subtree raising:* subtree B was removed from the tree and subtree C was raised to its position. The instances that were previously classified under 1 are now classified either 2 or 3.

The performance of both trees cannot be evaluated on the data that was used to build the original tree. The original tree was fit to this data and it would thus always outperform the pruned tree. A standard way to evaluate the trees is to split the original dataset to a training set used to build the trees (2/3 of data) and a validation set on which the error rates are estimated (1/3 data). This is known as a *holdout* procedure. If random noise affected the induction of the original tree, this will perform less well on new data because it is unlikely that it will contain the same kind of noise. The disadvantage of this approach is that it requires a large dataset in order to ensure that both subsets contain a complete range of data. Quinlan (1993, page 37) refers to this approach as "reduced error pruning".

To overcome this difficulty C4.5 (Quinlan, 1993, Chapter 4) attempts to estimate the error of trees on the training data itself using a heuristic based on statistics.[9] The observed error rate of a node is determined by considering instances N that reach that node and by assuming that their correct target class is the majority class at that node. Instances with other classes are errors E. The observed error rate is thus $f = E/N$. It is

[9]The approach is not entirely statistically valid because of the way it estimates the error rate from the training set, assumes that errors have a normal distribution and uses a 25% confidence interval to estimate the interval boundaries of the true error rate.

assumed that the occurrence of error among N instances follows a binomial/Bernoulli distribution which is used to estimate the true error rate. With a certain level of confidence, the true error rate lies within some interval around the observed error rate. The size of the interval is dependent on the number of observations. If N is large, the observed error rate will be close to the true error rate and hence the interval will be small. If N is small, the interval has to be increased to ensure the same confidence of estimation. A 95% confidence means the likelihood to find the true error rate in that interval. This value is standardly used in statistic estimation. C4.5 on the other hand uses a 25% confidence. This means that the confidence level in the predicted interval will be small but the interval will be narrower than with a 95% confidence. The upper limit of this interval is taken as a pessimistic estimate of the true error rate. There is no particular reason why 25% is chosen except that it ensures a satisfactory performance. Reducing this parameter encourages more pruning since smaller error is estimated.

How are the estimated error rates compared? In the case of subtree replacement shown in Figure 3.11 the error rate is first estimated individually for both leaves 2 and 3. Their combined estimated error rate is determined by taking the weighted sum of both estimated error rates. The weights are the proportions of examples that are classified under each leaf. The error rate of the parent node C is estimated by considering instances that have reached that node. The node C is replaced with a leaf representing the most frequently occurring class among instances reached that node if the estimated error rate of the node C is less than the combined estimated error rate of the leaves. Similar steps are taken to compare the error rates in subtree raising shown in Figure 3.12 except that here the combined estimated error rate of leaves 1, 2 and 3 in the first tree is compared with the combined estimated error rate of leaves 2 and 3 in the second.

3.5 Results

Understanding how classifiers work we can now turn to a discussion of their performance on our datasets. Our interest is twofold. We want to compare how the individual classifiers perform in our domain in general. Secondly, we want to compare their performance on the individual subsets of data from the domain. These subsets reflect different choices that we made while creating the system. We collected the data in two environmental settings (descriptions of robotic motion and descriptions of object relations), we controlled the complexity of linguistic description (*Simple* and *All* datasets), we used different instance creation methods (*Time-shifted* instances, *Zero-Time-shifted* instances and instances that were *Not-Time-shifted*, see page 73), and finally numeric attributes were discretised to different numbers of nominal categories. The estimated performance would tell us which of our choices is the optimal one.

Throughout we use the Weka implementations (Witten and Frank, 2005) of the algorithms discussed in the previous section with their default options. Naive-Bayes does not require any additional configuration. For J48 we can choose whether we want to use "reduced-error pruning" for which we can specify the size of the training and test datasets, or whether we want to estimate the error rates for pruning from the training dataset. We use the latter method with the default confidence level of 25% as suggested by Quinlan (1993). We can also specify whether subtree raising is used or not. By default this option is turned on. Finally, we can specify the minimum number of instances per leaf which controls the integration of noise. This value is set to 2. The command line options for J48 are thus `-C 0.25 -M 2`.

Before presenting and discussing the performance of classifiers in numbers we discuss different measures of performance.

3.5.1 Classifier evaluation measures

Evaluating the classifiers induced by machine learning is as important as creating them.[10] This is because of the problem of *over-fitting* to the dataset from which they were built. The classifiers are a theory about the training data. If the training data is unrepresentative of the data the classifier will encounter in the future either because the data is biased, deficient or contains errors, the classifiers will not perform well on the new data. Testing a classifier on the original dataset should hopefully give very few errors, also known as *re-substitution errors*. However, this figure will be over-optimistic of the true performance of the system, for example, when the robot is used in a new environment. In order to be a reliable estimate of the true system's performance, any performance figure that we come up with has to conform to the statistical criteria for making estimates from population samples.

If data is readily available we can collect three datasets under identical conditions. We use one for *training*, the second for *validation*, and the third for *testing*. The obtained level of error or accuracy is then applied to a statistical test which treats correct and incorrect classifications as a series of Bernoulli trials, to predict the confidence intervals which indicate how reliable the estimate is (for details see Witten and Frank, 2005, page 146,ff.). In this case the reliability of the estimate is solely dependent on the size of the test set.

Unfortunately, in practice the luxury of having three datasets is rarely possible. The information is scarce and the cost of obtaining information is high. To return to our setting: although the numeric data sampled from the robotic sensors is abundant, the amount of training data is limited by the number of linguistic descriptions that a human describer can produce in one session. Experience shows that in an hour session a human can generate approximately 100 descriptions. Human describers easily get

[10] We use the term "classifier" somehow loosely to refer to three related concepts: (i) the classification algorithm, (ii) its implementation, and (iii) the theory about the data that has been learned using this implementation.

bored or may not be prepared to devote extensive time to the experiment. But even if we have a reasonably sized dataset that can be split into two or three statistically representative subsets, we may nonetheless want to devote as much data as possible to training as this helps to induce the best possible model.

As discussed previously, a good compromise is the *holdout* procedure but for estimating the performance of classifiers rather than optimising them. We do not have to split the dataset into 2/3 for training and 1/3 for testing or estimate the true classifier performance on the basis of a single accuracy figure. The procedure can be repeated several times. The estimated classifier performance is taken as the mean performance over all iterations. At each iteration we randomly sample instances and place them in each subset. Since an instance is removed from the sampling set after it has been sampled it can never happen that the same instance would occur simultaneously in the training set and the test set. It may be the case that random sampling does not create representative subsets. The estimated classifier performance suffers if instances of a certain class are completely missing in either of the subsets. In parallel, this means that these instances are over-represented in the other subset which leads to a pessimistic estimate of the true performance. To ameliorate this situation we can use random sampling with *stratification* which ensures that equal proportions of the target class are represented in both subsets.

Random sampling from the previous method ensures that the same instance is not used for both training and testing in one run but it does not ensure that each instance is used for training and for testing *exactly once* across all the runs. If sampling is not random, the estimates from the iterations may be biased if certain instances are more likely to be chosen for testing than for training. *Cross-validation* is a special case of the holdout procedure where this bias is overcome. The instances for the training set and the test sets are not picked individually at each iteration. The procedure first splits the entire dataset into n partitions which are also referred to as *folds*. During each iteration

the classifier is trained on $n-1$ folds and tested on the remaining fold. The procedure is repeated n times and the estimate of true classifier accuracy is obtained by averaging the accuracies obtained in each fold. The most commonly used number of folds is 10 which is shown by experience to give good estimation results. The procedure is known as *stratified 10-fold cross-validation*.

Kohavi (1995) compares the cross-validation method with another method known as the 0.632 bootstrap method (for details see Witten and Frank, 2005, page 152 ff.) on a variety of real-world datasets using the C4.5 and Naive-Bayes classifiers. He considers the bias of error estimation methods such as pessimistic or optimistic and the variance of estimates between different runs. He concludes that a ten-fold stratified cross-validation provides the best estimates of error. This method is also the most commonly chosen method for testing classifiers and therefore we also use it here.

It is important to note that in Weka and also in other machine learning implementations the training step and the testing step are independent. At the end of each training and testing Weka prints out information which includes the model that was learned followed by various performance measures estimated from the stratified 10-fold cross-validation. Looking at this information it is easy to conclude that the models that are created in each fold of the validation are combined and the integrated model is output. This is not the case. The model that is output and which we subsequently used to generate new linguistic and non-linguistic behaviour of the robot is created on the entire training set. The model is optimised during a separate optimisation step. The models created in iterations of the testing step are approximations of the output model and their quality of approximation or their stability is dependent on the size and the structure of the dataset. After being used for testing in each iteration, these models are discarded. They are only used to gather statistics which are averaged and output. They are not integrated into the original model.

We discussed how the performance or the error rate can be statistically estimated

in the case of a limited dataset but we said nothing about what the measure of the performance may be. Assuming that a classifier predicts a nominal class, the most straightforward way to measure its performance is to count the number of times it predicts a correct value or the number of times it predicts an incorrect one. The count is subsequently divided by the total number of test classifications. In the first case we get classifier *accuracy* or *success rate*, in the second, we get its *error rate*.

The two rates are not always a good indicator of classifier performance. Figure 3.13 shows *confusion matrices* for two classifiers whose accuracy was estimated to be 99% and error rate 1%. A confusion matrix is a table in which the predicted target class values are cross-tabulated with the actual class values. Each cell contains the number of instances that had a particular predicted and actual value. The counts are sums from all 10 folds of a stratified 10-fold cross-validation and their total sum equals the number of instances in the dataset used for training the classifier.

		Predicted	
		yes	no
Actual	yes	50	0
	no	1	49

(a) Classifier A

		Predicted	
		yes	no
Actual	yes	99	0
	no	1	0

(b) Classifier B

Figure 3.13: A 99% accuracy or 1% error rate is not always a good indicator of performance.

When classifiers are trained and tested they treat all target class values equally. Consequently, all errors made on them are considered equal as well. Because training is optimised on these errors the classifiers that are built do not differentiate between different *costs* of predicting the wrong value. The target class values are not equal in their distribution in the training set. For example, Classifier A was trained on a set containing an equal proportion of "yes" and "no" classes. On the contrary, Classifier B was trained on a dataset where the majority of instances belonged to the class "yes". The confusion matrices built during the evaluation of the classifiers show that Classifier A

can predict both classes with a similar degree of accuracy and hence overall 99% accuracy is a good estimate of its performance, whereas Classifier B has a good accuracy on the "yes" class but fails completely when predicting the "no" class. Thus, the 99% accuracy is not a very informative measure of its actual performance. As demonstrated by Witten and Frank (2005, pages 161–162) with the dairy herd example, in practice we are often interested in predicting the less frequently occurring class.

Target class values are not equally important to us. Imagine a simplified situation in which we want to predict whether the robot should make a specific move or not. False predictions are necessarily associated with cost. Since the predicted target values are not equally important, the cost of predicting the wrong, in this case the opposite, value is also different. For example, if we wrongly predict that the robot should make a move (false positive) it may crash into a wall, whereas if we wrongly predict that it should not move (false negative) we may only end up with a frustrated user complaining that the system has not recognised their command. Different costs of incorrect predictions may also occur when predicting linguistic categories such as *Verb* or *Relation* but here the the cost is dependent on the lexical semantics of words belonging to these classes. For example, if the two words are exact synonyms ("going", "moving") or antonyms ("going", "stopped") then the cost of incorrectly predicting the opposite class is identical for making either error. In the first case the cost will be zero and in the second case the cost will be maximal. However, if there is hyponymy/hypernymy relation between the words where the words establish a synonymous reference only in a subset of cases ("forwarding", "going"), then the costs of predicting a different category will be different too.

If costs are known they can can be taken into account either at training or classification (see Witten and Frank, 2005, Section 7.5). In the present work we do not attempt to provide cost-sensitive learning or classification. Nonetheless, we will consider some measures of classifier *performance per class* calculated by the Weka evaluation compo-

nent which give us some idea what the costs would be when a particular classifier is used. All the estimates are calculated from a confusion matrix created in a stratified 10-fold cross-validation. Figure 3.14 shows a confusion matrix with labels that are commonly used to refer to different kinds of classifications.[11] If the target class has more than two values, the confusion matrix will be wider and deeper depending on the number of values. In this case, for the purposes of calculating the evaluation measures below, "positives" refer to instances belonging to the class in question and "negatives" refer to instances of all other classes.

		Predicted	
		yes	no
Actual	yes	true positives (TP)	false negatives (FN)
	no	false positives (FP)	true negatives (TN)

Figure 3.14: A confusion matrix showing different classification possibilities

The *Accuracy* (Equation 18) and the *Error rate* (Equation 19) are general measures of classifier performance on a particular target concept as discussed before. They are included here to show how they relate to other class-specific measures. The *TP Rate* (Equation 20) expresses the percentage of instances belonging to the class C that the classifier assigned correctly to the class C. On the other hand, the *FP Rate* (Equation 21) expresses the percentage of instances belonging to other classes that the classifier assigned to the class C. For example, if in a two-class situation the *FP Rate* for the class "yes" is 0.6, this means that the classifier is performing less well on the "no" class since more than half of the "no" instances were classified as "yes". Thus, a good classifier would have a high *TP Rate* and low *FP Rate* for every value of the target class. For a two class target concept the *TP Rate* of the class C and the *FP Rate* of the class ¬C sum to 1.

[11] In statistical literature false rejections (FN) are also known as Type I errors and false acceptances (FP) are known as Type II errors.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{18}$$

$$\text{Error rate} = 1 - \text{Accuracy} \tag{19}$$

$$\text{TP Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{20}$$

$$\text{FP Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{21}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{22}$$

$$\text{Recall} = \text{TP Rate} \tag{23}$$

$$\text{Fallout} = \text{FP Rate} \tag{24}$$

$$\text{F-Measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \tag{25}$$

TP Rate is sometimes referred to as *Recall* (Equation 23) and *FP Rate* is also known as *Fallout* (Equation 24). The term *Recall* originates in Information Retrieval (IR) where it is compared to *Precision* (Equation 22). Manning and Schütze (1999, Figure 8.1, page 268) illustrate the notion of *Precision* and *Recall* with a diagram in Figure 3.15. The *selected set* (FP + TP) includes the items that the retrieval method selected, the *target set* (TP + FN) represents the relevant items that should be selected. *Precision* and *Recall* are the ratios of TP in either of the sets: *Precision* is the proportion of TP in the selected set and *Recall* is the proportion of TP in the target set. *Precision* tells us how exact the results of the retrieval method are and *Recall* tells us how thorough or complete is the system returning the relevant items.

There is a trade off between the two measures. We get a high *Precision* and low *Recall* if the classifier is conservative and classifies only a subset of instances belonging to class C as class C. For example, we have a classifier which takes the coordinates of the located (LO) and reference (REFO) objects and finds the description of relation between them. If the classifier assigns the class "in front of" only to those instances where the located object is perfectly aligned with the reference object at 0 degrees north

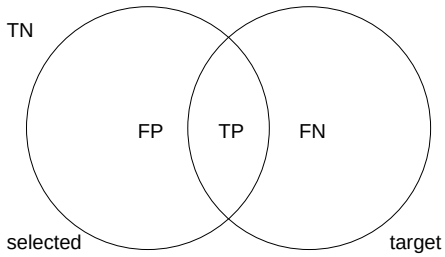


Figure 3.15: The relationship between *Precision* and *Recall*

relative to the origin of the robot and where $LO_y < REFO_y$, all such instances would be correctly classified as "in front of" and *Precision* would be 100%. However, such classifier would also create a lot of FNs which the *Precision* measure does not consider, instances that should be classified as "in front of" but did not match the strict criteria and were incorrectly identified as "to the right of" or "to the left of". As a result, the *Recall* would be low. Alternatively, a 100% *Recall* and a low *Precision* on the class "in front of" would result from a situation in which the classifier assigns all instances to this class. This is because this ensures that all instances belonging to this class are classified as this class. However, there would also be a lot of FPs, instances of other classes that were misclassified as this class. *Recall* does not take them into account but they are important for *Precision*. A good classifier would thus give us high *Precision* and high *Recall* for every value of the target class.

Precision and *Recall* can be combined in a single measure known as the *F-Measure* (Equation 25) which is attributed to van Rijsbergen (1979).[12] The *F-Measure* is a weighted harmonic mean of *Precision* and *Recall* which is just one of the ways to average numbers in mathematics (for details see Rennie, 2004). In the *F-Measure* we lose the trade off between FP and FN expressed by *Precision* and *Recall*. However, the *F-Measure* is still informative as it maximises with TP. It thus tells us how good the classifier is in

[12]Manning and Schütze (1999, page 269) point out that the *F-Measure* is a variant of the *E-Measure* from van Rijsbergen (1979) where $F = 1 - E$.

selecting that class.

The evaluation component of Weka calculates both sets of measures (*TP Rate/FP Rate* and *Precision/Recall*) on the basis of a stratified 10-fold cross-validation. We discuss these measures in respect to our classifiers in Section 3.5.6. We start with a discussion of the classifier accuracy.

3.5.2 Classifier accuracy

As discussed in Section 3.2 we collected two datasets: a developmental dataset called *Simple* which was collected under restricted conditions (one human describer, constrained unambiguous vocabulary, automated data entry method to prevent errors due to delay), and a full dataset called *All* where all of these conditions were relaxed. A classifier was trained on each dataset and for each target concept using both J48 and Naive-Bayes learning methods. The estimated accuracies of the resulting classifiers are shown in Table 3.7 and Figure 3.16.

Concept	Instances	J48	Naive-Bayes
	Simple		
Verb	82	89.02	78.05
Direction	82	87.80	78.05
Heading	82	97.56	98.78
Manner	82	70.73	71.95
Relation	278	75.90	80.58
	All		
Verb	1435	48.22	37.49
Direction	1435	55.68	44.46
Heading	1435	60.77	49.41
Manner	1435	54.70	45.57
Relation	625	69.12	67.20

Table 3.7: The estimated accuracies using stratified 10-fold cross-validation of 5 classifiers trained on *Simple* and *All* datasets

It can be seen straight away that both *J48* and Naive-Bayes perform better on the *Simple* dataset (marked in red) than on the *All* dataset (marked in blue). For example, the mean and the standard deviation for classifiers trained with the *J48* learner are

$\bar{x} = 84.20\%$, $s = 9.64$ for the *Simple* dataset, and $\bar{x} = 57.70\%$, $s = 6.97$ for the *All* dataset. The figures for Naive-Bayes are $\bar{x} = 81.48\%$, $s = 9.10$ (*Simple*) and $\bar{x} = 48.83\%$, $s = 9.96$ (*All*). The difference in performance of classifiers on both datasets is considerable and is expected. While collecting the *Simple* dataset we deliberately eliminated all factors that could have introduced external and thus undesirable irregularities to the dataset. We can thus treat the estimated accuracies from this dataset as accuracy upper bounds: they are the accuracies the two learners can achieve in an ideal situation conditioned only by the complexity of the learning task. An accuracy above 80% for the upper bound is an encouraging performance. However, the accuracies below 50% for the classifiers trained on the *All* dataset are less encouraging.

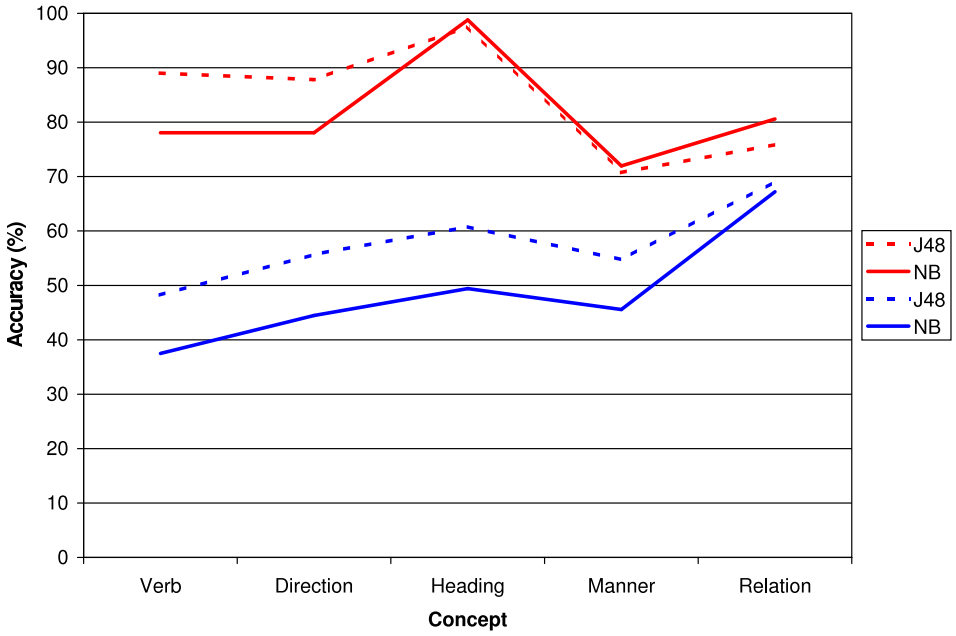


Figure 3.16: The estimated accuracies using stratified 10-fold cross-validation for 5 classifiers trained on *Simple* (red) and *All* (blue) datasets

The classifiers for the motion classes (*Verb*, *Direction*, *Heading* and *Manner*) built on the *All* dataset are performing considerably worse than the classifiers for *Relation* (J48 Motion: $\bar{x} = 54.84\%$, $s = 4.46$, J48 Relation: 69.12%; Naive-Bayes Motion: $\bar{x} = 44.23$, $s = 4.30$, Naive-Bayes Relation: 67.20%). One would assume that learning descriptions of object relations is a more difficult task than learning descriptions of motion as they

are semantically more complex. The difference in the classifier accuracy can thus perhaps be attributed to the problem of synchronising entries with observations of robotic motion and entries with natural language descriptions as discussed in Section 3.3.1. Here, all classifiers for the motion classes were trained on instances that were created without the time-shifting mechanism. We return to the discussion of classifier performance on datasets where this mechanism was introduced in the following section.

The four curves in Figure 3.16 follow a similar pattern which confirms that not all categories are equally easy to learn. For example, *Verb* appears to be a more difficult category to learn than *Heading*. Note that this is not linked to the number of the target classes of each concept: in the *All* dataset *Verb* has eight target classes and *Heading* has eleven.

Finally, Figure 3.16 shows that J48 classifiers perform better than those based on Naive-Bayes. Naive-Bayes only outperforms J48 on the *Simple* dataset for the *Relation* class by 4.68%. The improved performance of J48 over Naive-Bayes is most prominent in the case of the motion categories of the *All* dataset (around 10%), yet it is partially also observable on the *Simple* dataset. For the *Relation* category the difference in the estimated accuracy on the *All* dataset is small: only 1.92%. Overall, the results confirm that Naive-Bayes, a conceptually very simple method of machine learning, may perform comparably to other more complex systems.

3.5.3 Classifier accuracy excluding chance

Accuracy is expressed as the percentage of hits between the value predicted by the classifier and the actual value of a test instance. It is measured as *agreement* between the predicted value and the actual value. The standard measure of agreement used in computational linguistics is the *kappa coefficient* (κ) which was introduced by Cohen (1960) and later brought to the attention of the computational linguistics community by Carletta (1996). The coefficient was designed for measuring agreement between

coders in annotation tasks such as annotating dialogue and discourse structure. Because of high subjectivity involved in these tasks, there is a considerable variation in performance between different annotators and hence a measure that accounts for annotator bias was required.

A human annotator or a classifier may be biased by assigning the majority class to all instances or by assigning all target class values with identical probability distributions and still achieve good accuracy rates. However, such a classifier does not add any knowledge or benefit to classification since the same results could be obtained without it. The accuracies produced by these theoretical classifiers are commonly taken as baselines to which the accuracies of true classifiers are compared. By calculating the κ coefficient we include an implicit comparison of classifier performance to both of these baselines.

The basic principle behind the κ coefficient is expressed by the following equation:

$$\kappa = \frac{A_o - A_e}{1 - A_e} \tag{26}$$

where A_o is the *observed agreement* and A_e is the *expected agreement* or *agreement by chance*. It follows that $1 - A_e$ is the maximum agreement attainable excluding the agreement by chance and $A_o - A_e$ is the actual agreement obtained excluding the agreement by chance. The range that κ can take is between $-A_e/1 - A_e$, when there is no observed agreement, and 1, when there is a perfect observed agreement.

A few different versions of the agreement coefficient are offered in the literature all of which follow Equation 26. Different proposals differ along two dimensions: (i) whether they calculate the agreement for two coders or multiple coders, and (ii) in the method by which the expected agreement A_e is estimated. Some coefficients also take into account the extent of disagreement between a particular pair of categories. For a review of different coefficients see Artstein and Poesio (2005). The evaluation component of Weka calculates the κ coefficient as proposed in Cohen (1960).

The expected agreement must be estimated from the observed data because we do not know the independent prior distributions of items to categories. We assume that when coders assign tags, they do so independently. It follows that the probability of two coders agreeing by chance is $P(k|c_1) \times P(k|c_2)$ where c_1 and c_2 stand for coders and k is the category or the tag or class assigned. The expected agreement is the probability of coders agreeing on any category or

$$A_e = \sum_{k \in K} P(k|c_1) \times P(k|c_2) \tag{27}$$

Cohen's κ assumes that each annotator follows a unique distribution of items into categories and that items are not assigned to categories uniformly. Therefore, to calculate the expected agreement A_e, we need to find for each coder c, the classifier and the human describer(s) who produced the dataset, the number of assignments to k or n_{ck} and dividing it by the number of all assignments i by that coder.

$$P(k|c) = \frac{n_{ck}}{i} \tag{28}$$

The κ coefficient is rarely used to compare the performance of classifiers in machine learning. Ben-David (2007) argues that it should be adopted instead of accuracy since it is a better indicator of the classifier merit and since different (statistical) conclusions about the performance of classifiers may be reached depending on which measure is considered. His study compares the performance of five well-known classifiers from Weka including *J48* and *Naive-Bayes* on fifteen datasets. The results show that on average about 35% of successful classifications measured as accuracy are random hits. The ranking of 5 classifiers according to accuracy and according to κ was different in 8 out of 15 datasets. This is because some classifiers are more likely to generate random hits than others even when they are used on the the same dataset.

Table 3.8 shows the estimated κ values for the *J48* and *Naive-Bayes* classifiers trained on both datasets. These are graphically represented in Figure 3.17. Following Ben-

Concept	κ(J48)	RA	Rκ	κ(Naive-Bayes)	RA	Rκ
			Simple			
Verb	0.7738	1	1	0.5699	2	2
Direction	0.7758	1	1	0.5170	2	2
Heading	0.9479	2	2	0.9744	1	1
Manner	0.2523	2	1	0.2102	1	2
Relation	0.6747	2	2	0.7379	1	1
			All			
Verb	0.2852	1	1	0.1639	2	2
Direction	0.2960	1	1	0.1846	2	2
Heading	0.3432	1	1	0.1978	2	2
Manner	0.0859	1	1	-0.0098	2	2
Relation	0.6110	1	1	0.5877	2	2

Table 3.8: The κ values ranked by the learning method

David (2007) we first rank classifiers according to the learning method – *J48* and *Naive-Bayes* – which gives us ranks 1 and 2 expressed across the table columns. RA indicates the classifier's rank relative to accuracy (given in Table 3.7) and Rκ indicates its rank relative to κ. Rank 1 indicates the best performance. As shown in Table 3.8 the classifiers rank identically relative to accuracy or κ in all but one case: when they are built from the *Simple* dataset and when classifying for the concept *Manner*. Therefore, using κ rather than accuracy does not change our conclusion that in the majority of cases *J48* performs better than *Naive-Bayes*.

A finer comparison of both measures can be made by examining ranks according to the concept learned which gives us ranks from 1 to 5 expressed across the rows of Table 3.9. There are very few differences in the classifier ranking on the *Simple* dataset: only in the case of the classifiers for *Verb* and *Direction*. Here the ranking of the two consecutive classifiers is reversed (J48: ranks 2 and 3; Naive-Bayes: ranks 3 and 4). There is also very little difference between the accuracy and κ values for items with consecutive ranks (J48 κ: 0.002, J48 accuracy: 1.22%, Naive-Bayes κ: 0.0529, Naive-Bayes accuracy: 0%).

Concept	*J48*		*Naive-Bayes*	
	RA	Rκ	RA	Rκ
	Simple			
Verb	2	3	3.5	3
Direction	3	2	3.5	4
Heading	1	1	1	1
Manner	5	5	5	5
Relation	4	4	2	2
	All			
Verb	5	4	5	4
Direction	3	3	4	3
Heading	2	2	2	2
Manner	4	5	3	5
Relation	1	1	1	1

Table 3.9: The ranks of the κ values by concept

When classifiers were trained on the *All* dataset, the estimated performance of *J48* is different according to the measure under consideration for concepts *Verb* and *Manner* which change ranks 4 and 5. Here the difference between the values of consecutive ranks is greater (J48 κ: 0.1993, J48 accuracy: 6.48%). The greatest variation in ranks is observed in the case of *Naive-Bayes* trained on the *All* dataset where 3 out of 5 concepts get ranked differently depending on the measure being used. Thus, the conclusions made by Ben-David (2007) are also replicated by our results.

One of the most important advantages of comparing the performance of classifiers with κ rather than accuracy is that with κ a majority based or random classifier always scores a 0 value and thus any value of κ that we cite contains an implicit comparison of that classifier to a baseline classifier. As shown in Figure 3.17, for the classifiers that were trained on the *Simple* dataset the average value of κ is high (J48: $\bar{\kappa} = 0.68$, $s = 0.23$, Naive-Bayes: $\bar{\kappa} = 0.60$, $s = 0.25$), whereas for those trained on the *All* dataset the value of κ is considerably lower (J48: $\bar{\kappa} = 0.32$, $s = 0.17$, Naive-Bayes: $\bar{\kappa} = 0.22$, $s = 0.20$). As with accuracy, an antisymmetry between the motion concepts and the *Relation* concept is observed. The greatest variation in the κ values is observed with the classifiers for

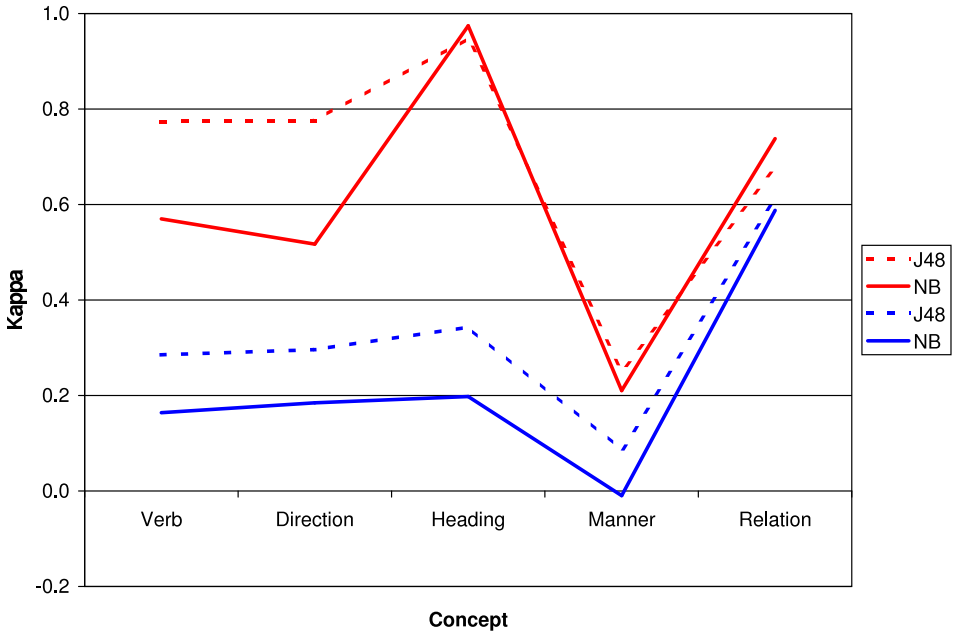


Figure 3.17: The performance of classifiers according to κ trained on *Simple* (red) and *All* (blue)

the *Motion* concepts (Simple J48: $\bar{\kappa} = 0.69$, $s = 0.26$, All J48: $\bar{\kappa} = 0.25$, $s = 0.10$, Simple Naive-Bayes: $\bar{\kappa} = 0.57$, $s = 0.27$, All Naive-Bayes: $\bar{\kappa} = 0.13$, $s = 0.08$), whereas the values for the *Relation* concept are much more similar (Simple J48: $\kappa = 0.6747$, All J48: $\kappa = 0.6110$, Simple Naive-Bayes: $\kappa = 0.7379$, All Naive-Bayes: $\kappa = 0.5877$). The lowest κ values are obtained on the *Manner* concept. Taking an average of classifiers based on both classification methods we get Simple: $\bar{\kappa} = 0.2313$ and All: $\bar{\kappa} = 0.0761$.

The size of the κ coefficient only has a meaning in relation to the task that we are evaluating, it is relative to the difficulty of the task. Artstein and Poesio (2005) discuss the various cut-off points that have been proposed in the literature. Among these the most influential in computational linguistics is the one proposed by Carletta (1996). There, κ is primarily intended for content analysis where levels of $\kappa > 0.8$ are considered as good reliability indicators, while values that fall within $0.67 < \kappa < 0.8$ are those from which only tentative conclusions can be drawn. On the other hand, in medical

research where κ originated from, levels as low as 0.4 are considered to be adequate. Artstein and Poesio say that their own experience shows that in annotation tasks a substantial agreement is obtained at around 0.7 level, but only values above 0.8 assure a reasonable quality annotation. On the other hand, they claim, that even the lower 0.67 level has often been impossible to achieve.

In the view of Artstein and Poesio (2005), κ should not be seen as an absolute performance measure cited alongside other experimental results but a measure that can give us an important insight into the task that we are investigating. Knowing that the true agreement of annotators in a coding task is beyond chance is trivial, since the aim of the task is to ensure a resource of reasonable quality which is used in further processing. The values of κ are dependent on how the categories used in annotation are defined and how well they fit the underlying phenomenon that the coding task is trying to record. If a value of 0.5 is obtained for a tagging task, then the tag-set that we are using does not model linguistic data very well, there is just too much disagreement between the annotators, and hence the dataset may not be of a sufficient quality for further processing.

For evaluating classifiers though, a value of 0.5, or in fact any value above 0, is a good result. It tells us that a classifier performs considerably better than a baseline classifier and thus its usage and the cost of construction can be justified. In this respect, all but one of our classifiers perform well. The κ value of the Naive-Bayes classifier for *Manner* is -0.0098 which means that the classifier is showing a minor (1%) disagreement with the dataset and thus in this case we were better off using a random classifier.

A relative comparison of κ values for classifiers for different concepts tells us how difficult is that concept to learn. The number of target class values and their distribution in the dataset should be excluded from such comparison and hence they are discounted in the calculation of the agreement by chance. Classifiers based on both

classification methods are struggling with *Motion* concepts when trained on the *All* dataset, especially with the *Manner* attribute. This seems to be a particularly difficult attribute to learn since the performance on it is also relatively low when the classifiers are trained on the *Simple* dataset (Manner κ: 0.2523 (J48), 0.2102 (Naive-Bayes), Other $\bar{\kappa}$: 0.7931 (J48), 0.6998 (Naive-Bayes)). The shape of the curves for κ in Figure 3.17 is different than the shape of the curves for accuracy in Figure 3.16 where the worst performance is estimated on the *Verb* concept. However, the conclusions about the general performance of classifiers relative to accuracy are also confirmed relative to κ. The concepts of *Motion* are much more difficult to learn from the *All* dataset than from the *Simple* dataset. The performance on the *Relation* concept is similar between the datasets. Its κ value is a very encouraging result. We hypothesised that the decreased performance on the concepts of *Motion* is due to the noise that was introduced during instance creation. The figures discussed here were obtained on instances for which the time-shifting algorithm was not applied. We return to the discussion of classifier performance on motion concepts created from instances where the time-shifting algorithm was applied in the following section.

Finally, let us consider what the κ coefficient tells us about the classifier performance on each value of the target concept. In a two class situation where there is an uneven distribution of values ($n_{yes} = 95$ and $n_{no} = 5$), the assignments of items belonging to "yes" to "no" have a different effect on the agreement by chance (A_e) than the assignment of "no"s to "yes". Let us consider a classifier that incorrectly assigns 3 members of "no" to "yes" and vice versa and calculate the agreement by chance for each case as shown in Equations 30 and 31.

$$A_e = P(\text{yes}|\text{dat}) \times P(\text{yes}|\text{class}) + P(\text{no}|\text{dat}) \times P(\text{no}|\text{class}) \quad (29)$$

$$A_{e1} = \frac{95}{100} \times \frac{98}{100} + \frac{5}{100} \times \frac{2}{100} \quad (30)$$

$$= 0.93$$

$$A_{e2} = \frac{95}{100} \times \frac{92}{100} + \frac{5}{100} \times \frac{8}{100} \quad (31)$$
$$= 0.88$$

In the first case the 3 misclassified items are multiplied as a proportion of 100 by the proportion 95/100 rather than 5/100. The resulting κ values that we obtain are considerably different: $\kappa_1 = 0.56$ and $\kappa_2 = 0.75$. Note that in both cases the observed agreement/accuracy (A_o) is the same: 0.97. If we assign more "no"s to "yes", the value of A_e converges to 0.95 when all "no"s are assigned to "yes" ($95/100 \times 100/100 + 5/100 \times 0/100$), and if we assign more "yes"s to "no", it converges to 0.05 when all "yes"s are assigned to "no" ($95/100 \times 0/100 + 5/100 \times 100/100$). Thus, in the first case the A_e is increasing whereas in the second case it is decreasing with the number of the opposite assignments. This is intuitively sound: the more items of a minority class are assigned to the majority class the less sophisticated is the assignment method. On the other hand, the more items of the majority class are assigned to the minority class, the more sophisticated is the method since such assignments cannot be attributed to chance.[13]

Since increasing agreement by chance results in lower values of κ, this measure provides a cost sensitive method of estimation of classifier performance. The errors made on a minority class are penalised heavier or are more costly than the errors made on the majority class.

3.5.4 Improving classifier performance on motion classes

In Section 3.3.1 we described a time-shifting procedure for combining information from MOOS log entries into instances in attempt to reduce synchronisation errors that arose between the components of the MOOS system and human describers when sampling

[13]Of course in both cases the observed agreement/accuracy also approaches the same extremes which means that the overall value of κ will approach 0. This is because such a classifier is a majority based classifier.

descriptions of robotic motion. The results from the previous section confirm that classifiers performed less well on the concepts of motion in comparison to the relation concept. In this section we examine whether the time-shifting procedure has a positive effect on the creation of motion instances by reducing noise.

As described on page 73 the procedure allows us to create two datasets which we call *Time-shifted* (TS) and *Zero-Time-shifted* (ZTS). The latter is a subset of the former and contains instances for which the time-shifting procedure determines that there is no delay between natural language and environment descriptions. We compare the performance of classifiers built from these datasets with those built from the datasets that were created without time-shifting *Not-Time-shifted* (NTS). The performance of the latter was discussed in the previous section. As before, we distinguish between *Simple* and *All* configurations which means that we are comparing classifiers built from six different datasets which are as follows: *Simple-Not-Time-shifted* (82), *All-Not-Time-shifted* (1435), *Simple-Time-shifted* (83), *All-Time-shifted* (1446), *Simple-Zero-Time-shifted* (51), *All-Zero-time-shifted* (586). The numbers indicate the number of instances in each dataset. The reason why *Not-Time-shifted* and *Time-shifted* datasets have a different number of instances is because the time-shifting procedure creates a virtual "stopped" description which corresponds to the last entry of a log file. The estimated accuracies from a stratified 10-fold cross-validation for both J48 and Naive-Bayes classifiers built from these datasets are shown in Table 3.10.

In order to compare the performance of classifiers built from various datasets we introduce a measure which we refer to as "relative difference" (RD). This is a difference between the estimated performance of a classifier built from the improved dataset (either *Time-shifted* or *Zero-Time-shifted*) and the baseline dataset (*Not-Time-shifted*) relative to the the performance of the classifier built from the baseline dataset as shown in Equation 32. The relative difference thus gives us the percentage by which the performance of a classifier has increased or decreased by introducing the time-shifting

Concept	J48			Naive-Bayes		
	NTS	TS	ZTS	NTS	TS	ZTS
			Simple			
Verb	89.02	92.77	98.04	78.05	92.77	100.00
Direction	87.80	72.29	80.39	78.05	61.45	80.39
Heading	97.56	85.54	96.08	98.78	74.70	100.00
Manner	70.73	68.67	70.59	71.95	73.49	66.67
			All			
Verb	48.22	57.12	56.83	37.49	55.88	52.56
Direction	55.68	64.59	72.35	44.46	41.56	39.76
Heading	60.77	65.63	67.75	49.41	45.78	43.00
Manner	54.70	56.22	63.99	45.57	54.15	42.15

Table 3.10: Classifier accuracies for the motion concepts on three datasets

procedure.

$$\text{RD} = \frac{(\text{TS}|\text{ZTS}) - \text{NTS}}{Abs(\text{NTS})} \tag{32}$$

Following the conclusions from the previous section, we use the κ coefficient as a better estimate of classifier performance.[14] If non-optimal synchronisation of information during creation of instances introduced randomness to the dataset, this randomness can only be captured by a random or a majority based classifier. Since κ does not favour such classifiers, an improvement in its estimated values gives us a better indication whether some randomness has been removed. Table 3.11 shows the estimated κ coefficients for the J48 classifier and Table 3.12 for the Naive-Bayes classifier. The relative differences RD-TS and RD-ZTS from the last two columns are represented graphically in Figures 3.18 and 3.19 respectively. As before red lines indicate *Simple* datasets and blue lines indicate *All* datasets.

Considering J48 first (Figure 3.18) we can see that there is an antisymmetry in the performance of classifiers built from the *Simple* and *All* datasets when the time-shifting procedure is applied. The κ coefficient improves on average by $\bar{x} = 64.22\%$,

[14] κ values can be negative and hence Equation 32 must include the *Abs* function in its denominator.

Concept	NTS	TS	ZTS	RD-TS	RD-ZTS
			Simple		
Verb	0.7738	0.8552	0.9602	10.52	24.09
Direction	0.7758	0.4086	0.5923	-47.33	-23.65
Heading	0.9479	0.6781	0.9253	-28.46	-2.38
Manner	0.2523	0.0078	-0.0268	-96.91	-110.62
			All		
Verb	0.2852	0.4139	0.4313	45.13	51.23
Direction	0.2960	0.4466	0.5665	50.88	91.39
Heading	0.3432	0.4470	0.4585	30.24	33.60
Manner	0.0859	0.1981	0.1668	130.62	94.18

Table 3.11: The performance of J48 classifiers according to κ on motion concepts on three datasets

$s = 39.07$ in the case of the *All* dataset. A high standard deviation indicates that the relative improvement in the value does not affect all concepts equally. The concept that most benefits from the time-shifting is *Manner*. It changed from $\kappa = 0.0859$ to $\kappa = 0.1981$ which is particularly encouraging (130.62%). The average improvement in performance of classifiers built from the *All-Zero-Time-shifted* subset is very similar: $\bar{x} = 67.60\%$, $s = 25.96$.

Contrary to expectations, the time-shifting procedure considerably decreases the performance of all J48 classifiers except for the *Verb* if it is applied on the *Simple* dataset (*Simple-Time-shifted:* $\bar{x} = -40.55\%, s = 38.66$ and *Simple-Zero-Time-shifted:* $\bar{x} = -28.14\%$, $s = 50.53$). Intriguingly, Figure 3.18 shows that the pattern of improvement is reversed when compared to the one observed with the *All* dataset. The decrease in performance is the highest on the *Direction* (*Simple-Time-shifted:* -47.33%, *Simple-Zero-Time-shifted:* -23.65%) and the *Manner* concept (*Simple-Time-shifted:* -96.91%, *Simple-Zero-Time-shifted:* -110.62%).

The time-shifting algorithm therefore does not perform well on the *Simple* dataset. The dataset contains a lot of descriptions "stopped", 36 out of 82 or 43.90%, which

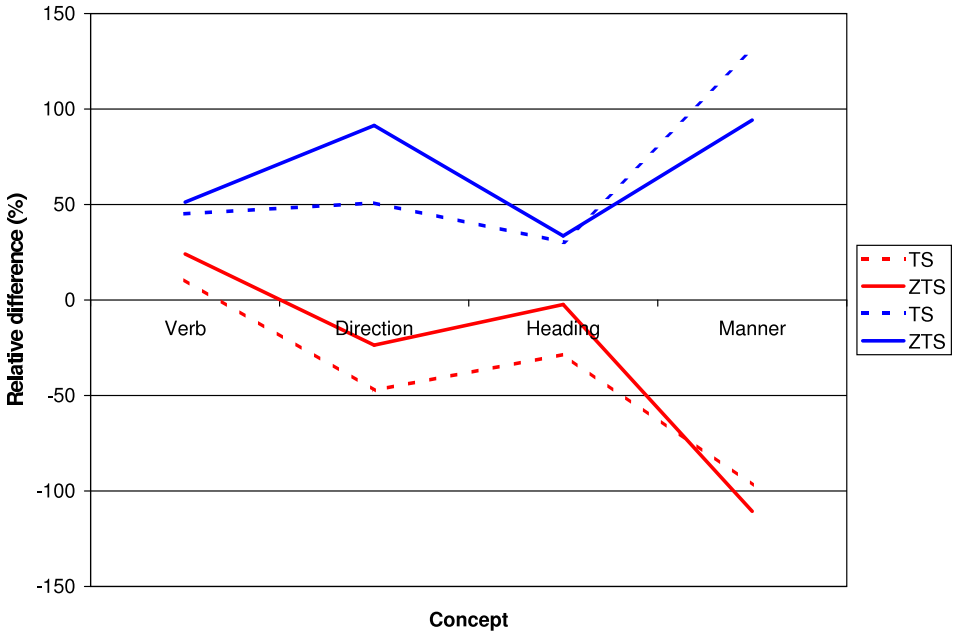


Figure 3.18: The relative difference in κ contributed by time-shifting for the J48 classifiers trained on *Simple* (red) and *All* (blue)

are used to delimit the dataset into segments on which time-shifting is performed (see Section 3.3.1). In comparison, in the *All* dataset the description "stopped" occurs in 203 out of 1435 instances, thus in 14.15%. In the *Simple* dataset each segment thus on average contains only one description other than "stopped". The more descriptions "stopped" there are, the more likely it is that these too are not synchronised with the motion of the robot. The time-shifting algorithm calculates a delay for each segment by subtracting the time code when the vehicle is stationary, when its speed is 0, from the time code when the vehicle is described to be stationary. Under normal circumstances natural language descriptions temporally follow descriptions of the robot's state. The algorithm regresses to find a stationary robot as far as it can on the list of odometry time codes not checking whether it encounters another preceding description "stopped". In some cases, especially if there are a series of descriptions of "stopped" which do not refer to a completely stationary robot, the regression may proceed too far and conse-

quently the estimated delay is too large. This causes the descriptions of a segment to be shifted back too far and matched with completely random odometry information. A manual investigation of the *Simple* dataset revealed that this indeed was happening.

The time-shifting procedure is not perfect and it could be improved in two ways. The regression to find an odometry entry describing a stationary robot could be limited not to cross into the preceding segment, it should not be allowed to cross a description "stopped". If information about a stationary robot cannot be found in any particular segment, the segment should either not be time-shifted or it should be discarded. Another solution would be to relax the condition for a stationary robot. Instead of looking for zero speed, an arbitrary small number such 0.03 could be used. Whichever heuristic we take, it may improve the creation of instances in certain situations but equally it may also create new errors in others. The preceding discussion shows that time-shifting improves the performance of the J48 classifiers based on the *All* dataset which was the reason why it was introduced. We attributed the impoverished performance of the classifiers built from the time-shifted *Simple* dataset to the fact that it contains an unusual number of "stopped" descriptions and that the criteria for a stationary robot may be too strong. Normally time-shifting would not be used on this dataset because we do not expect it to contain dis-synchronised entries.

Concept	NTS	TS	ZTS	RD-TS	RD-ZTS
			Simple		
Verb	0.5699	0.8552	1.0000	50.06	75.47
Direction	0.5170	0.1371	0.5923	-73.48	14.56
Heading	0.9744	0.3854	1.0000	-60.45	2.63
Manner	0.2102	0.2156	0.0586	2.57	-72.12
			All		
Verb	0.1639	0.3680	0.3607	124.53	120.07
Direction	0.1846	0.1623	0.1426	-12.08	-22.75
Heading	0.1978	0.1763	0.1600	-10.87	-19.11
Manner	-0.0098	0.0436	-0.0148	544.90	-51.02

Table 3.12: The performance of Naive-Bayes classifiers according to κ on motion concepts on three datasets

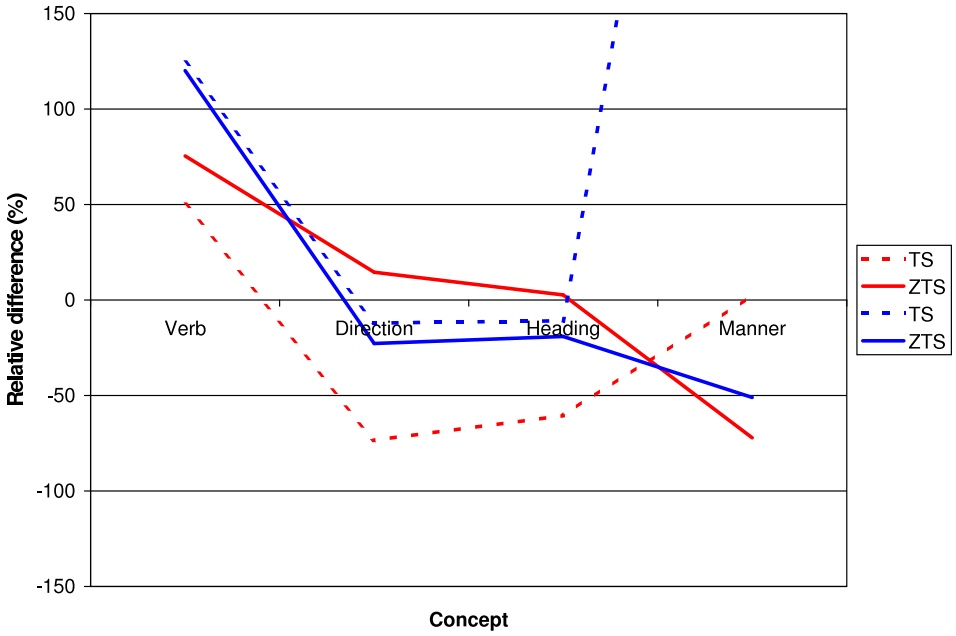


Figure 3.19: The relative difference in κ contributed by time-shifting for the Naive-Bayes classifiers trained on *Simple* (red) and *All* (blue)

The relative difference in performance of Naive-Bayes on various datasets is more varied (Figure 3.19) but the same overall picture emerges as with J48. On average time-shifting improves the performance of classifiers based on the *All* dataset but not those based on the *Simple* dataset (*Simple-Time-shifted:* $\bar{x} = -20.33\%$, $s = 49.78$; *All-Time-shifted:* $\bar{x} = 161.62\%$, $s = 228.15$). On *All-Time-shifted* the performance only improves in the case of *Verb* and *Manner*. The improvement on the latter amounts to 544.90% but in an absolute value this means only an increase of 0.05. The κ value for this concept is now positive rather than negative when no time-shifting is applied. The time-shifting makes the κ values for *Direction* and *Heading* worse by 12.08% and 10.87%. The performance of Naive-Bayes on *All-Zero-Time-shifted* improves less ($\bar{x} = 6.80\%$, $s = 66.56$) than on *All-Time-shifted* and this is surprising. The classifier for *Verb* is the only classifier whose performance improves on *All-Zero-Time-shifted*. Notably the classifier for *Manner* gets worse rather than improves (-51.02%). Perhaps the *All-Zero-Time-shifted* subset contains sparse data which performs less well in machine learning. However, on the *Simple* dataset the situation is almost reversed. The classifiers for *Verb*, *Direc-*

tion and *Heading* improve more on *Simple-Zero-Time-shifted* than on *Simple-Time-shifted*. However, the classifier for *Manner* gets considerably worse (-72.12%).

Although both J48 and Naive-Bayes were trained on identical datasets time-shifting appears to affect them quite differently. It gives rise to a less predictable improvement in performance in the case of Naive-Bayes compared to J48. Furthermore, as shown by the contours of the lines in Figures 3.18 and 3.19, in the case of Naive-Bayes the relative difference, whether positive or negative, is much more extreme. The time-shifting improved the κ value of both learners on all datasets for the *Verb* concept. It had the greatest effect, either positive and negative, on the *Manner* category.

Should time-shifting be used or not? The preceding discussion shows that it has mostly a positive effect on the performance of classifiers if it is applied when creating instances from the *All* dataset. However, it probably should not be used on the *Simple* dataset.

3.5.5 How many nominal classes from numeric attributes?

In Section 3.3.4, page 74 we mentioned that we had to discretise numeric attributes when they were used as target concepts with J48 and Naive-Bayes classifiers to nominal attributes with classes representing intervals of real numbers. The number of nominal classes had to be determined manually and doing so there is a trade off between the accuracy of the classifier and its predictive performance. To find the optimal number of classes for each numeric attribute we conducted an experiment in which we split our six numeric attributes *Delta-Heading* (*DH*), *Speed*, *LO_x*, *LO_y*, *REFO_x* and *REFO_y* into various numbers of intervals or bins as described on page 75 and then compared the performance of J48 and Naive-Bayes on such target concepts. The baseline for our comparison was the mean performance of the same classifiers used on the same datasets but on concepts that were truly nominal.

Throughout, we consider the version of the *Simple* dataset where the instances for the motion concepts were created *without* time-shifting (*Simple-Not-Time-shifted*) and

the version of the *All* dataset where the instances for the motion concepts were created *with* time-shifting (*All-Time-shifted*). When these datasets are considered together with the datasets for object relations which were never created with time-shifting, we refer to them only as *Simple* and *All*.

The discretisation in which some number of bins is created from an attribute whose values are real numbers biases the classifiers toward one of the two extremes as discussed in Section 3.3.4. This is because we do not know what are meaningful categories for this attribute. These depend on the values of other attributes in the dataset. On one hand, if only one bin is created, the classifier will inevitably behave like a majority based classifier. On the other hand, if too many bins are created, for example, so that each numeric class fits into its own bin, then the classifier will behave like a random classifier. It is thus here too that a comparison of κ on various datasets will be more instructive than a comparison of accuracy. We nonetheless start with a brief comparison of the various classifiers by accuracy.

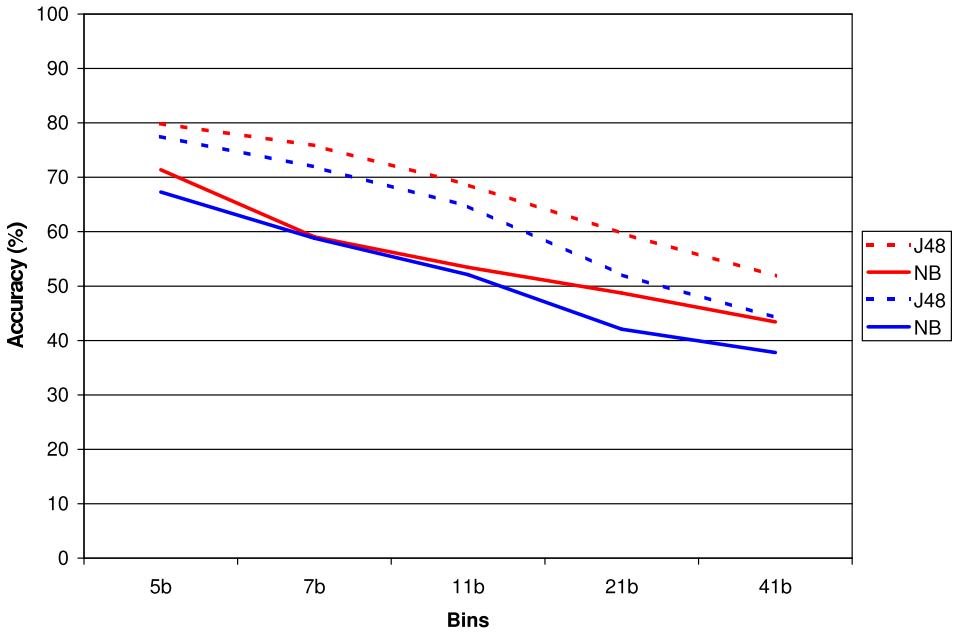


Figure 3.20: The average accuracies of classifiers with differently discretised real-valued target concepts trained on *Simple* (red) and *All* (blue). Each bin is a nominal category of a target concept and represents an interval of values from the set of real numbers.

Figure 3.20 shows how the average accuracies for both J48 and Naive-Bayes classifiers from *Simple* and *All* datasets taken from Table 3.13 change while altering the number of bins of target concepts. As predicted in Section 3.3.4, the accuracies decrease with the number of bins, simply because with an increased number of classes it becomes more difficult for the classifier to place an instance in the correct class. The mean accuracy on truly nominal concepts for *Simple J48* is $\bar{x} = 84.20$, $s = 9.64$ and for *Simple Naive-Bayes* $\bar{x} = 81.48$, $s = 9.10$. As shown in Table 3.13, the closest mean accuracies on this dataset are already reached with our smallest chosen number of bins (5). We mark their values in Table 3.13 in bold. On the other hand, the mean accuracies on nominal concepts for *All J48* are $\bar{x} = 62.54$, $s = 5.03$ and for *All Naive-Bayes* $\bar{x} = 52.91$, $s = 8.88$. For these the closest mean accuracies are obtained if real-valued target concepts are split into 11 bins.

		5b		7b		11b		21b		41b	
Concept	Inst	J48	NB	J48	NB	J48	NB	J48	NB	J48	NB
					Simple						
DH	82	86.59	87.80	81.71	80.49	78.05	79.27	71.95	74.39	67.07	68.29
Speed	82	70.73	80.49	74.39	79.27	67.07	63.41	58.54	58.54	52.44	47.56
LO_x	278	67.27	60.43	61.51	20.50	48.56	16.91	34.53	11.51	28.42	11.15
LO_y	278	79.50	34.17	78.06	27.70	68.35	22.66	52.16	17.27	32.73	14.39
REFO_x	278	84.17	78.42	71.58	64.75	64.75	61.51	61.87	56.83	61.15	54.32
REFO_y	278	91.01	87.05	87.77	81.65	84.89	76.98	79.14	73.74	69.42	64.75
Mean		**79.88**	**71.39**	75.84	59.06	68.61	53.46	59.70	48.71	51.87	43.41
Standard deviation		8.47	18.94	8.24	25.43	11.35	24.72	14.28	25.24	16.03	22.71
					All						
DH	1444	82.89	78.88	73.48	72.23	69.11	66.90	53.53	52.84	45.64	46.05
Speed	1444	64.89	64.96	58.86	58.24	49.58	48.13	36.29	35.66	29.16	28.19
LO_x	625	65.44	63.84	60.16	50.24	48.80	40.00	34.08	16.80	28.96	17.28
LO_y	625	80.32	37.28	82.08	33.76	72.80	26.88	57.28	23.52	37.44	18.24
REFO_x	625	81.44	80.00	71.52	63.84	65.60	60.80	58.88	57.76	58.88	55.20
REFO_y	625	89.92	78.72	85.28	74.56	82.24	69.92	72.32	65.92	65.28	61.76
Mean		77.48	67.28	71.90	58.81	**64.69**	**52.10**	52.06	42.08	44.23	37.79
Standard deviation		9.23	14.97	9.94	13.87	12.08	15.33	13.28	18.05	13.94	17.52

Table 3.13: The accuracies of classifiers with differently discretised real-valued target concepts trained on *Simple* and *All*

An interesting picture emerges when we look at the corresponding means of κ values shown in Table 3.14 and Figure 3.21. For the *Simple J48* run the highest average κ coefficient is obtained with 7 bins, for *Simple Naive-Bayes* with 5 bins, for *All J48* with

11 bins and for *All Naive-Bayes* with 7 bins. Three of four runs (*Simple J48*, *All J48* and *All Naive-Bayes*) show an arching curve with a peak between 7 and 11 bins. Thus, if a choice on the number of bins should be made on the basis of Figure 3.21, 7 bins are optimal, although a very rough, choice.

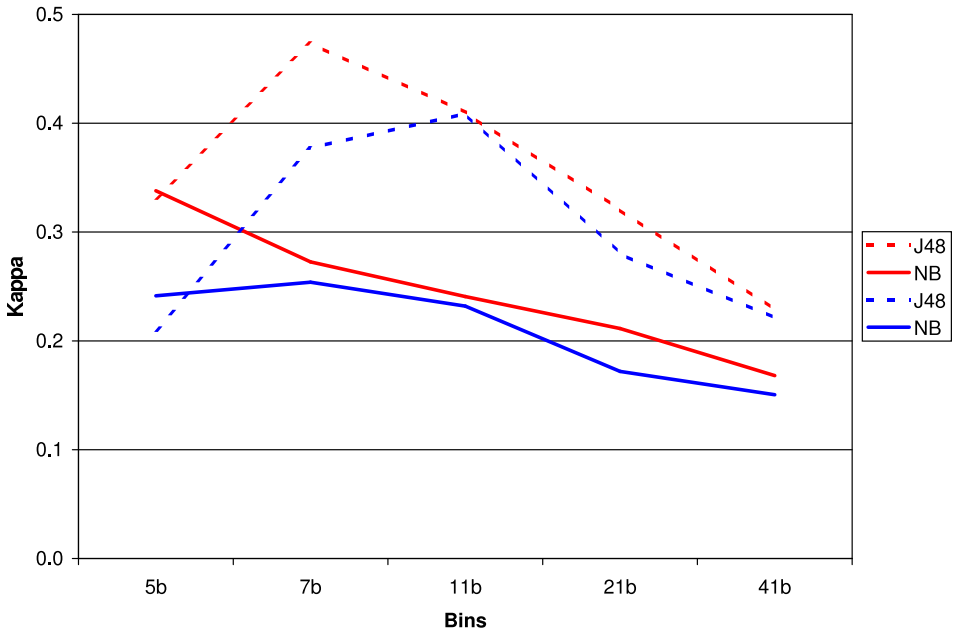


Figure 3.21: The average values of κ of classifiers with differently discretised real-valued target concepts trained on *Simple* (red) and *All* (blue)

How do the average κ values of classifiers with differently discretised real-valued target concepts compare to the average κ values of classifiers with truly nominal target concepts? When discretisation is applied, the average κ values of classifiers based on *Simple* are considerably lower. The mean κ for *Simple J48* on nominal target concepts is $\bar{x} = 0.6849$, $s = 0.2335$ and $\bar{x} = 0.6019$, $s = 0.2526$ for *Simple Naive-Bayes*. As shown in Table 3.14 the closest average values of κ for J48 with discretised target concepts are $\bar{x} = 0.4735$, $s = 0.1464$ for 7 bins and for Naive-Bayes $\bar{x} = 0.3380$, $s = 0.2738$ for 5 bins. These are also the highest average values for all bins under consideration. The mean κ for *All J48* on nominal target concepts is $\bar{x} = 0.4233$, $s = 0.1320$ and $\bar{x} = 0.2676$, $s = 0.1909$ for *All Naive-Bayes*. The performance of classifiers with discretised target concepts comes close to these values (*All J48:* $\bar{x} = 0.4090$, $s = 0.0846$ for 11 bins; *All*

Naive-Bayes: $\bar{x} = 0.2539$, $s = 0.1219$ for 7 bins). Again, these are the highest values for all the bins under consideration. The results show that the discretised classes are not as semantically straightforward as the natural nominal classes of the *Simple* dataset. However, the quality of the discretised classes is comparable to the more varied classes of the *All* dataset which is a positive result.

	5b		7b		11b		21b		41b	
Class	J48	NB	J48	NB	J48	NB	J48	NB	J48	NB
					Simple					
DH	0.6979	0.7251	0.6168	0.5920	0.5679	0.5904	0.4535	0.4999	0.3921	0.4138
Speed	0.4532	0.6568	0.5659	0.6590	0.4711	0.4366	0.3792	0.3891	0.3045	0.2315
LO_x	0.4842	0.3833	0.4703	0.1099	0.3274	0.0796	0.2309	0.0590	0.2116	0.0722
LO_y	0.0000	0.0372	0.5750	0.0823	0.4792	0.0665	0.3259	0.0509	0.1838	0.0705
REFO_x	0.3765	0.1592	0.1773	0.0848	0.1274	0.1003	0.1602	0.1396	0.0029	0.0851
REFO_y	-0.0234	0.0664	0.4354	0.1072	0.4859	0.1711	0.3732	0.1294	0.2762	0.1357
Mean	0.3314	**0.3380**	**0.4735**	0.2725	0.4098	0.2408	0.3205	0.2113	0.2285	0.1681
St.dev.	0.2615	0.2738	0.1464	0.2505	0.1449	0.2006	0.0980	0.1711	0.1211	0.1232
					All					
DH	0.0000	0.2722	0.0818	0.3692	0.4456	0.3932	0.2809	0.2831	0.1913	0.2243
Speed	0.4786	0.4731	0.4396	0.4262	0.3737	0.3481	0.2548	0.2299	0.1869	0.1551
LO_x	0.4464	0.4023	0.4410	0.3187	0.3396	0.2328	0.2191	0.0999	0.2079	0.1225
LO_y	0.0000	0.0605	0.6437	0.1229	0.5301	0.0922	0.3735	0.0926	0.2325	0.0888
REFO_x	0.3376	0.1317	0.2241	0.1310	0.2839	0.1560	0.2426	0.1642	0.2357	0.1495
REFO_y	-0.0024	0.1088	0.4326	0.1554	0.4812	0.1697	0.3094	0.1619	0.2711	0.1625
Mean	0.2100	0.2414	0.3771	**0.2539**	**0.4090**	0.2320	0.2801	0.1719	0.2209	0.1505
St.dev.	0.2151	0.1543	0.1792	0.1219	0.0846	0.1070	0.0506	0.0675	0.0291	0.0412

Table 3.14: The values of κ of classifiers with differently discretised real-valued target concepts trained on *Simple* and *All*

In the preceding discussion we compared the average performance of classifiers for various numbers of bins across all discretised concepts. However, it is quite likely that different concepts will have a different optimal number of bins, depending on our conceptualisation of motion and space and thus related to the linguistic descriptions. Table 3.15 shows the number of bins with the highest scoring value of κ for both learners and on both datasets. An interesting fact that emerges from this table is that except in one case (*All*, REFO_x) both J48 and Naive-Bayes achieve the best performance with the same number of bins. This gives weight to the claim that there is an optimal number of bins for a particular discretised numeric target concept. Although there is some correspondence, the number of bins for one target concept is not the same between

Simple and *All* datasets. This can be expected as the datasets do not contain the same nominal attributes which drive the classification to the discretised classes of the target concept.

	DH	Speed	LO_x	LO_y	REFO_x	REFO_y	Mean
				Simple			
J48	5	7	5	7	5	11	7
NB	5	7	5	7	5	11	5
				All			
J48	11	5	5	7	5	11	11
NB	11	5	5	7	21	11	7

Table 3.15: Bins with the highest κ value per concept

No doubt the success of discretisation is dependent on the way other attributes split the data (see Witten and Frank, 2005, Figure 7.4, page 303). An unsupervised method of finding the optimal number of bins for a numeric concept discussed in Witten and Frank (2005, Section 7.2, page 298) is *entropy-based discretisation* which was introduced by Fayyad and Irani (1993). The approach is almost identical to the way numeric attributes are split into leaves by the C4.5 algorithm described in Section 3.4.2. It requires that the target concept is a nominal attribute different from the numeric attribute to be discretised. The numeric attribute is discretised to bins that give rise to the purest subsets of instances in respect to the values of the target concept. This is accomplished by sorting the values of the numeric attribute from the lowest to the highest and then considering each point where the target class changes (see Table 3.6 on page 90). For each possible split an information value is calculated, and the split point that gives rise to the lowest information value is the one where an interval split is introduced.[15] The procedure is further recursed on both intervals until some stopping criteria is reached.[16]

[15]This ensures that the information gain which is the difference in information value before the split and with the split is the greatest. The information value before the split is a constant and can thus be omitted.

[16]This is determined in terms of the Minimal Description Length Principle (MDL) by considering whether the information required to specify the split (the theory) and the target classes according to that split (the data) is smaller than the information gain introduced by the split. For discussion see (Witten and Frank, 2005, page 301).

Although this method is superior to our manual method based on comparing the final classifier performance in terms of the κ coefficient we have not implemented it. This is because we started the task as a practical examination of the effects of different bin sizes on the performance of classifiers during which reasonable results were obtained. No doubt, future work should include the implementation of the entropy-based discretisation.

3.5.6 The performance of classifiers per class

In this section we examine the performance of classifiers per class. As discussed in Section 3.5.1, there are two measures that are commonly used for this purpose: the ratio of true positives (*TP Rate*) to false positives (*FP Rate*) and the ratio of *Precision* against *Recall*. Since all our concepts are multi-valued rather than binary, negatives refer to all other classes excluding the class under consideration. Each ratio is calculated for every target class of a concept and the pairs of ratios are cross-tabulated in a *ROC Space* or a *Precision-Recall Graph*. Because of the limitations of space we cannot consider each of eleven target concepts in detail. Since they were built on two datasets and we are examining them for two measures, this would result in 44 graphs in total. Instead, we chose to discuss the performance of classifiers on two representative target concepts only—*Relation* and *Verb*—and list the evaluation data for all other concepts in Appendix A, page 255ff.[17]

All ratios are determined by examining a confusion matrix produced in a stratified 10-fold cross validation such as the one in Figure 3.22. The matrix shows us what errors are made by the classifier. For example, the class *g* "in the front of" was classified correctly 84 times. However, instances belonging to this class were also classified as *i* "to the right of" (15), as *j* "to the left of" (13), as *l* "close to" (7), as *a* "next to" (1), as *f* "facing" (1), as *h* "far from" (1), and as *k* "behind" (1). Equally, instances of the class

[17]This also includes the calculation of *AUC* (Area Under the ROC Curve) for each target class which we omit in our discussion. For details see (Witten and Frank, 2005, page 168ff.).

j "to the left of" were classified correctly 131 times but they were also classified as *g* "in the front of" (17), as *k* "behind" (8), as *i* "to the right of" (4), and as *c* "near" (3). The examples show that the majority of incorrect classifications are on classes whose denotations or spatial templates partially overlap ("to the left of" versus "in the front of" versus "to the right of") as discussed on page 15. For this reason they are also very hard to distinguish for the classifier.

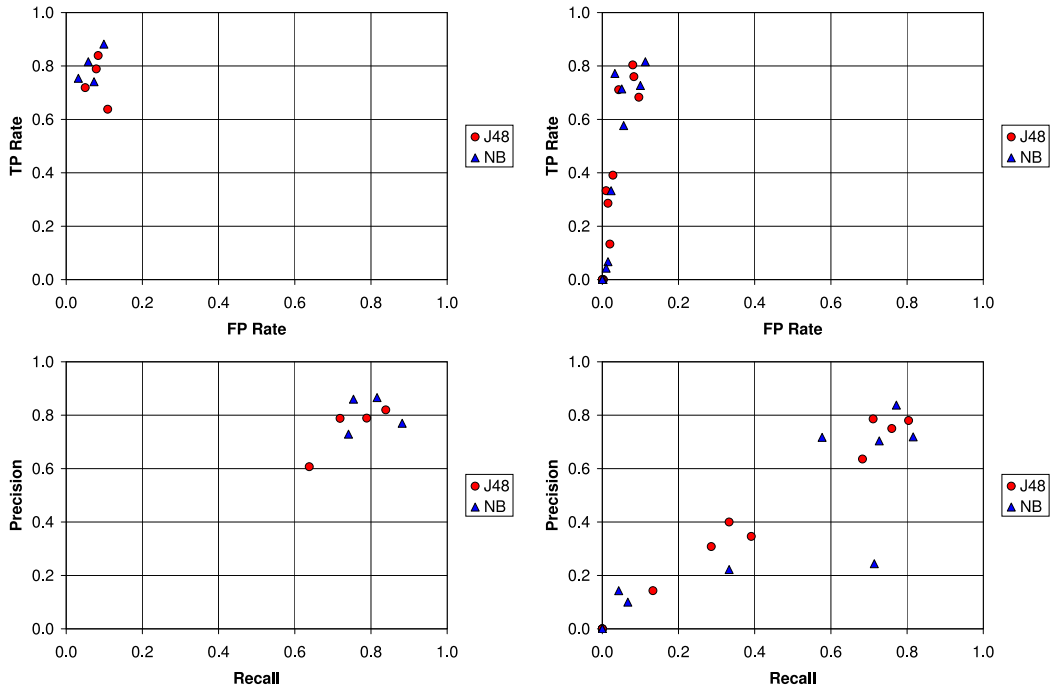
	Predicted												
	a	b	c	d	e	f	g	h	i	j	k	l	
	0	0	0	0	0	0	1	0	0	0	0	0	a "next to"
	0	0	0	0	0	0	0	0	0	1	0	0	b "after"
	0	0	2	0	0	0	1	0	2	5	0	5	c "near"
	0	0	0	0	0	0	0	0	2	0	1	1	d "parallel to"
	0	0	0	0	0	0	0	0	0	1	0	0	e "opposite of"
Actual	0	0	0	0	0	4	1	4	0	0	2	1	f "facing"
	1	0	0	0	0	1	84	1	15	13	1	7	g "in the front of"
	0	0	0	0	0	2	2	4	2	2	0	2	h "far from"
	0	0	3	0	0	0	17	1	117	7	9	0	i "to the right of"
	0	0	3	0	0	0	17	0	4	131	8	0	j "to the left of"
	0	0	2	0	0	3	7	0	13	7	81	1	k "behind"
	0	0	4	2	0	0	2	3	1	1	1	9	l "close to"

Figure 3.22: A confusion matrix for the J48 classifier for the concept *Relation* built by a stratified 10-fold cross-validation from the *All* dataset

Figure 3.23 shows the ROC Space and the Precision-Recall graph for the *Relation* classifier built from both *Simple* and *All* datasets. As concluded previously, the classifiers based on the *Simple* dataset are performing well. The points representing the individual target classes cluster in the top left corner of the ROC graph and in the top right corner of the Precision-Recall graph. There is a high *TP Rate* or *Recall* and a high *Precision*. Importantly, the points representing individual target classes are clustered. This means that the classifiers are performing equally well on all target classes.

For the classifiers built on the *All* dataset, both the ROC graph and the Precision-Recall graph show two clusters. The classifiers perform well on some classes but less well on the others. The better performing cluster appears to correspond to the cluster found with classifiers based on the *Simple* dataset. We return to this issue below.

The ROC graph shows that the classes with a *TP Rate* below 0.4 also have a low



(a) Simple dataset

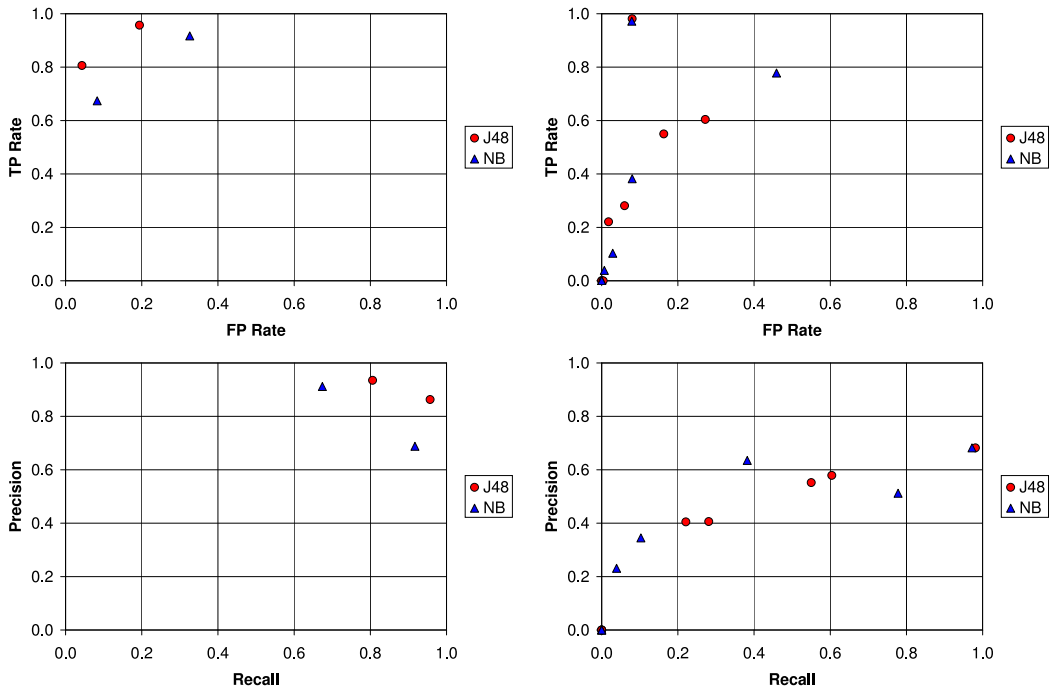
(b) All dataset

Figure 3.23: ROC and Precision-Recall graphs for *Relation*

or zero *FP Rate*. This means that the majority of instances belonging to these classes (0.6 or more) are assigned to other classes and that very few instances of other classes are classified as these classes. The Precision-Recall graph shows that both *Precision* and *Recall* appear to have a similar value below 0.4. This means that in the set of instances that are assigned a particular class only 40% or less truly belong to that class and that only 40% or less of that class is retrieved in the set.

The most likely explanation for this behaviour of classifiers on these classes is data sparseness. The classifiers were trained on a training set that contained only a few instances with each target class. For example, the confusion matrix in Figure 3.22 shows that "near" only occurs in 15 out of 625 instances and only 2 have been correctly assigned. On the other hand "next to" only occurs once in the entire dataset which means that it could not be represented in each training and testing set. A few instances are

insufficient to separate these classes from other classes. As a result they tend to be assigned other classes.

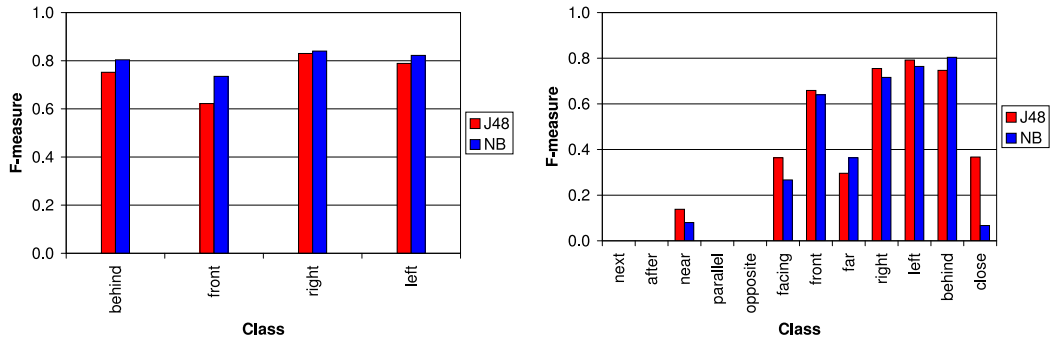


(a) Simple dataset (b) All dataset

Figure 3.24: ROC and Precision-Recall graphs for *Verb*

Figure 3.24 shows the same measures graphed for the concept *Verb*. The classifiers perform slightly worse on this concept (see Figure 3.17 on page 112 for a comparison of their κ values): the dots tend to move further away from the top left (ROC graph) and the top right corners (Precision-Recall graph). Also, the clusters are slightly less pronounced but the general pattern of graphs is strikingly similar to the pattern for the *Relation* concept.

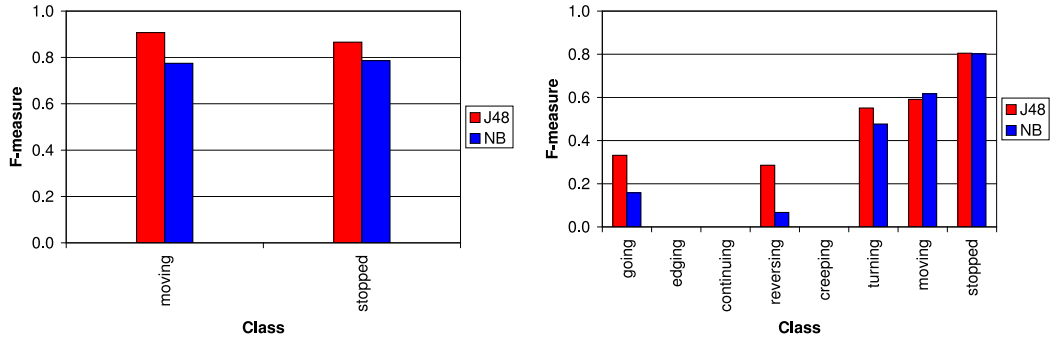
We discussed the general performance of classifiers per individual classes but we did not mention which classes are associated with each pattern. Figure 3.25 and 3.26 show histograms with class labels on x-axis and their *F-Measure* on y-axis. As discussed on page 103, *F-Measure* is an average of *Precision* and *Recall* which means that it allows us to represent them both conveniently on a single axis.



(a) Simple dataset (b) All dataset

Figure 3.25: F-Measure per class of *Relation*

Both sets of histograms confirm that the highest *F-Measures* are found with those classes of concepts built from the *All* dataset that were also chosen as the "core" classes for the *Simple* dataset. These are "behind", "in the front of", "to the right of" and "to the left of" for *Relation* and "moving" and "stopped" for *Verb*.



(a) Simple dataset (b) All dataset

Figure 3.26: F-measure per class of *Verb*

In the preceding sections we concluded that the classifiers perform less well on the *All* dataset than the *Simple* dataset. However, the evidence presented here shows that the fall in performance is not uniform with every value of the target class. The performance of classifiers on the "core" classes is comparable regardless which dataset they were built from. This is an encouraging result since it shows that varying the

describers while collecting the *All* dataset did not influence the learning of the "core" target classes. The diminished performance appears to be due to scarceness of other classes. This was an inevitable outcome of the experiment in which we wanted the describers to use freely any vocabulary that they deemed appropriate. It is interesting from a linguistic point of view that most describers usually opt for a "core" vocabulary to refer to the environment rather than display creativity. This is most likely because of a practical point: projective relations such as "to the left of" define the location of objects far more precisely than non-projective ones such as "close".

3.6 Conclusion

In this chapter we described how we collected, structured and performed learning on our two corpora of spatial expressions which we referred to as *Simple* and *All*. The *Simple* corpus was created by constraining the data collection process with the view of creating the cleanest possible dataset given the complexity of the task. In contrast, the *All* corpus was collected by relaxing these restrictions by allowing multiple describers, different experimental scenes, un-controlled use of linguistic descriptions and by providing a more natural way of describing the scenes through speech recognition. Our objective was to find associations between a relatively low level information that is sufficient to navigate a mobile robot and a higher level symbolic human conceptualisation of motion and space reflected in natural language descriptions.

While preparing the data for machine learning we had to make a number of choices. For example, we had to choose the kind of information that will be included in learning, how this will be structured and how information collected in different scenes shall be integrated. Each of these choices adds positive or negative knowledge to the learning task and thus biases the learning process. Another challenge that we faced is how to combine various non-linguistic and linguistic information from time stamped entries in MOOS log files to coherent instances. This was particularly important for creating

a corpus of motion descriptions since the values of entries were constantly changing and there was a high chance that the information would become dis-synchronised. We devised a time-shifting procedure which attempts to compensate such errors. Finally, another challenge was to find a suitable discretisation method to transform continuous numeric attributes to nominal classes that could be used with our selection of classifiers.

Two popular machine learning algorithms were chosen for the task: a decision tree learner J48 and Naive-Bayes classifier. Although Naive-Bayes is conceptually a much simpler classification method, it is commonly reported to achieve comparable results to other state of the art methods. This was also confirmed by our experiments. The performance of the classifiers was evaluated with a stratified 10-fold cross-validation and by considering different evaluation measures. Of these, the accuracy is the most common measure. We pointed out that the accuracy alone cannot be a conclusive measure of the classifier performance and chose the κ coefficient in preference to it. Here again, we confirmed the observations in the literature that different conclusions about the performance of the classifiers may be drawn depending on the measure being considered.

Throughout the chapter we compared the performance of the system in respect to the choices that were made: the classifiers, the use of time-shifting on the subset of the motion data and the optimal number of bins to discretise numeric attributes. The general trend that emerged from the evaluation is that the learners performed better on the *Simple* rather than the *All* dataset, that J48 performed better than Naive-Bayes and that time-shifting improved the performance of classifiers on motion data but only if this was taken from the *All* dataset. We argued that the optimal single overall number of nominal bins for numeric attributes is 7. By considering a detailed performance of classifiers per class for some target concepts we have shown that the performance of *All* may be comparable to *Simple* on values that were better represented in the dataset

and that the overall diminished performance may be due to data sparseness on some target class values.

Throughout this chapter we discussed the performance of the classifiers relative to the machine learning task. But can we also say that the learning was successful in general? How well do the classifiers approximate to the human conceptualisation of motion and space? No doubt the performance of any natural language system depends on the difficulty of the task and thus directly comparing our results to other systems, even if they concentrate on learning or generating spatial and motion descriptions, does not make much sense. For example, in a parsing task an accuracy of 98% is a good achievement. However, in automatic induction of grammar rules from text an accuracy of 60% is an excellent result. A much better test of learning would be to examine whether the learned knowledge can be used to generate actions and descriptions that appear natural to humans. We constructed a set of experiments toward this end which we describe and discuss in Chapter 5. Before proceeding to these we first discuss the construction of two systems which integrate the classifiers to basic language generation and question answering systems with which human observers can interact.

Chapter 4

From classifiers to an interactive system

4.1 Introduction

In this chapter we describe how we integrated the classifiers trained in machine learning to a simple language generation (*pDescriber*) and question answering system (*pDialogue*). The purpose of these systems is not to build a state of the art dialogue systems but to test the performance of classifiers on human observers. The learning tasks described in Chapter 3 can only be judged successful if the robot is able to use and understand the acquired expressions in a way that is natural to a human observer.

The systems integrate the classifier knowledge in two different ways. *pDescriber* most faithfully replicates the scenes in which the data was collected. The programme considers the odometry and SLAM information about the environment and produces natural language descriptions of various categories from which grammatical sentences are generated. *pDialogue*, on the other hand, generates answers to user's questions about the location of objects and performs motion actions based on user's commands. Both tasks require a series of steps in order to generate a response. For example, to answer a question about the location of objects we need to choose the relevant reference object in the scene and find a relation between that object and the object we want to locate. Equally, to generate a requested motion the robot must be issued with commands that bring the robot to the state referred to in the request. In terms of classifiers,

the difference between *pDialogue* and *pDescriber* is that in *pDialogue* in many cases the target concepts are not natural language descriptions but properties of the robot or environment.

To communicate with other robotic applications *pDescriber* and *pDialogue* must integrate with the MOOS environment. We discuss how this is done in the following section.

4.2 General structure of *pDescriber* and *pDialogue*

Both *pDescriber* and *pDialogue* communicate with the MOOS system through a wrapper programme called *iLinguistics*.[1] *iLinguistics* acts as a proxy between the MOOS system and the linguistic applications. These send and receive data to *iLinguistics* by writing and reading it from structured input and output text files, whereas *iLinguistics* sends and receives this data to the MOOS database through the TCP/IP protocol. This is summarised in Figure 4.1. This set up may not be optimal, but it was chosen because linguistic applications were designed in Prolog (Bratko, 2001), whereas MOOS applications are written in C++. The MOOS system already contains libraries that handle data exchange between MOOS applications and the MOOS database. Creating a simple wrapper MOOS application was simpler and faster than re-implementing or porting these libraries to Prolog.

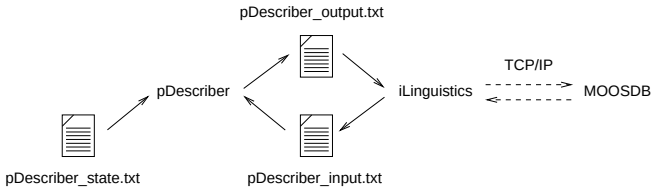


Figure 4.1: Integration of linguistic applications with MOOS

iLinguistics runs a linguistic application at a predefined interval. At each run *iLinguistics* retrieves the values of the desired variables from the MOOS database, formats

[1] I thank Paul M. Newman for implementing this application.

them to a particular form and writes them to the input file for the linguistic applications (*pDescriber_input.txt*). It then runs the linguistic application which reads and parses the data from the input file, performs the required actions using the data that it received, formats and saves the data that it wants to send to MOOS to the output text file (*pDescriber_output.txt*) and exists. The output file is then read by *iLinguistics* and the values are submitted as MOOS variables to the MOOS database. A shortcoming of this approach is that a linguistic application must exit before the data is read by *iLinguistics*. This means that linguistic applications are run in sessions. During each session a linguistic application is reinitialised. In order to keep track of the actions that happened before and also to read-in some settings particular to the current configuration and environment we use a special "state" file (*pDescriber_state.txt*). This contains feature-value pairs, one per each line. Figures 4.2 and 4.3 show two sample files used in the evaluation of both systems. Most entries have a straightforward interpretation. We will explain the others in the forthcoming sections when we discuss the functions of the systems. Just as the input and output files, the state file is read and parsed when a linguistic application is launched and it is updated with new information just before a linguistic application exits.

Through *iLinguistics* the linguistic applications can read and write the value of any variable that is published in the MOOS database. The messages exchanged between the applications and *iLinguistics* are strings containing feature-value pairs. We implemented two predicates *send_string(+Variable,+Value)* and *send_numeric(+Variable,+Value)* which format our data to the format expected by *iLinguistics*.[2] New variables can be created by simply sending a new variable name with some value. MOOSDB keeps track of the application that set the variable's value. In our case the application name is *iLinguistics*.

[2]According to a convention in Prolog programming the + symbol with an argument name indicates that the argument must be specified when the predicate is queried, – indicates that the argument is returned by the predicate, and ? indicates that the argument must either be specified or is returned.

```
vehicle_name                  =  MARGE
maximum_speed                 =  0.6
max_angular_velocity          =  1
robot_z                       =  0.5
room_x                        =  9.393
room_y                        =  11.97
odometry_source               =  iPlatform
objects_commentary_frequency  =  3
use_moos_ispeech              =  no
ask_for_judgements            =  yes
direction                     =  j48|weka.classifiers.trees.J48|all_ts_direction_j48.model...
heading                       =  j48|weka.classifiers.trees.J48|all_ts_heading_j48.model...
manner                        =  j48|weka.classifiers.trees.J48|all_ts_manner_j48.model...
verb                          =  j48|weka.classifiers.trees.J48|all_ts_verb_j48.model...
preposition                   =  j48|weka.classifiers.trees.J48|all_ts_j48.model...
delta_heading                 =  none
speed                         =  none
lo_x                          =  none
lo_y                          =  none
refo_x                        =  none
refo_y                        =  none
said_previously               =  stopped none none none
```

Figure 4.2: A state file that configures *pDescriber* for a particular environment and instructs it to use the J48 classifiers built from the *All-Time-shifted* dataset.

```
vehicle_name                  =  MARGE
use_moos_voice_input          =  yes
use_moos_speech               =  no
repeat_input_sentence         =  yes
room_x                        =  15.173
room_y                        =  11.586
robot_z                       =  0.5
must_confirm_motion           =  yes
ask_for_judgements            =  yes
binary_judgements             =  yes
direction                     =  j48|weka.classifiers.trees.J48|all_ts_direction_j48.model...
heading                       =  j48|weka.classifiers.trees.J48|all_ts_heading_j48.model...
manner                        =  j48|weka.classifiers.trees.J48|all_ts_manner_j48.model...
verb                          =  j48|weka.classifiers.trees.J48|all_ts_verb_j48.model...
preposition                   =  j48|weka.classifiers.trees.J48|all_preposition_j48.model...
delta_heading                 =  j48|weka.classifiers.trees.J48|all_ts_delta_heading_11b_j48.model...
speed                         =  j48|weka.classifiers.trees.J48|all_ts_speed_11b_j48.model...
lo_x                          =  j48|weka.classifiers.trees.J48|all_lo_x_11b_j48.model...
lo_y                          =  j48|weka.classifiers.trees.J48|all_lo_y_11b_j48.model...
refo_x                        =  j48|weka.classifiers.trees.J48|all_refo_x_11b_j48.model...
refo_y                        =  j48|weka.classifiers.trees.J48|all_refo_y_11b_j48.model...
heard_previously              =  Where is the sofa?
evaluate_response             =  no
response_in_session           =  -1
```

Figure 4.3: A similar state file to configure *pDialogue*.

As discussed in Section 2.4.1, page 31ff. *iLinguistics* does not receive and pass on all the data from the MOOSDB but only what is interested in. This is accomplished by subscribing to variables which can be initiated from a linguistic application using the *subscribe(+Variable)* predicate. The subscription to MOOS variables must be performed as soon as the system is launched, when the session count is 0.[3] If the session is other than 0, a linguistic application should already be receiving the information that it subscribed to and and it should proceed to processing it and issuing a response. Figure 4.4 shows Prolog pseudo-code which demonstrates the general framework of linguistic applications.

```
start :-
   get_command_line_arguments,
   read_data_from_files,
   main_application,
   write_data_to_files,
   reset_application.
```

```
main_application :-
   session_count(0),
   subscribe(Var1),
   subscribe(Var2),
   ...,
   .

main_application :-
   p_describer.

main_application.
```

Figure 4.4: General structure of linguistic applications. Very roughly, a colon followed by a hyphen (:-) can be read as *if* and a comma (,) can be read as *and*. Multiple definitions of the *main_application/0* predicate indicate alternative goals that are explored. This happens if the goal in the first definition fails, for example if *session_count(1)*. In this case the goal in the second clause is explored and so on.

Remember that MOOSDB does not delete any information once it has been retrieved by an application. The information remains there until it is specifically altered. This means that a client application may receive the same variable value each time it queries the database if this has not changed in the meantime. This is intended by design. MOOSDB records states of environment and the robot. If the state has not changed, the response of the system should also not change. However, this behaviour

[3]The session count is passed from *iLinguistics* to the linguistic applications as a command line argument together with the names of the required input and output files. See *get_command_line_arguments/0* in Figure 4.4.

may be problematic when considering linguistic descriptions which are acts of speech rather than states. For example, *pDialogue* receives linguistic descriptions through a MOOS variable called *VOICE_INPUT*. This is published by *iSpeech*, a programme providing an interface to a speech recogniser. When considering *VOICE_INPUT pDialogue* must check whether its value is not the same as in the previous session which is recorded in the *heard_previously* feature in the state file. If it is, it can be ignored since the human interactor said nothing new.[4]

The linguistic applications interface with Weka in a similar way as they interface with the MOOS system: through text files. Although writing a glue code that would interface with Java classes that make up Weka is easier than interfacing with the MOOS system, the simplicity of the text file data exchange outweighed these options. Surprisingly, the method provides a very good speed of classification.[5] In Weka a classifier can be saved after training in a model file that can be reloaded later to classify new instances. This step can be done by running the following command: *java weka.classifiers.trees.J48 -T to-classify.arff -p 5 -l learned.model*.

To make a successful classification the following information is required: the name of the classifier (*weka.classifiers.trees.J48*), the file with the model (*learned.model*), an arff file with instances to classify (*classify.arff*), and the index of the attribute that is the target concept (*5*). The arff file must contain a properly formatted header declaring attributes and their values as when performing training on them (see Figure 3.9, page 73). This also includes the target concept. Unless we are testing the performance of the classifier on a test set, the target classes of individual instances will be unknown. In such cases they are replaced by a question mark (*?*) which is commonly used in Weka datasets to designate unknown values.

[4] An alternative solution would be for *pDialogue* to set the value of *VOICE_INPUT* to some dummy value after processing a valid linguistic description and not provide any response in the subsequent sessions until a different value is encountered.

[5] The longest time in the pipeline is taken by the speech recogniser and later by a speech synthesiser. Nonetheless, the overall speed of the system is more than satisfactory and the system behaves quite naturally.

```
0 left 0.75 ?
1 right 0.7272727272727273 ?
2 far 0.95 ?
3 left 0.8813832578243578 ?
4 close 0.9843674373825883 ?
```

Figure 4.5: Weka classifier output

When running the classification command above the output in Figure 4.5 is obtained. Each line represents one classification. The number identifies instances in the arff dataset that were classified. The second property is the class predicted by the classifier model, followed by the prediction accuracy or confidence with which that class was chosen. The final property is the actual target class of the instance if this is given in the supplied dataset. This example shows that quite a few pieces of information from the training are required to classify new instances using an existing classifier model. We show in the following sections how the continuity of this information is ensured in our system.

4.3 Configuring and starting the systems

Figures 4.6 and 4.7 show the configurations of processes that are required to run each system. As expected, they are very similar to the configurations under which the data was collected shown in Figure 3.1, page 57. The difference is that the systems collect and process both motion and object relation data simultaneously and that the language input methods have been replaced by intelligent systems that are able to generate and analyse linguistic data rather than just collect it. Since the data was collected certain technical changes have been made to the MOOS system. Most notably, *iAGV* has been replaced by *iRelayBox* and *iPlatform* which now provide odometry information.

The difference between the configuration for *pDescriber* (Figure 4.6) and *pDialogue* (Figure 4.7) is that the latter includes another process *pDialogueEval*, a virtual user used

for the evaluation which asked predefined questions at specific locations which were then answered by the system. This process is redundant if the system is running in a non-evaluation mode and would be replaced by *iSpeech*, an interface to the speech recognition system to which a real user can ask a question. The boxes encompass processes that were running on one computer.

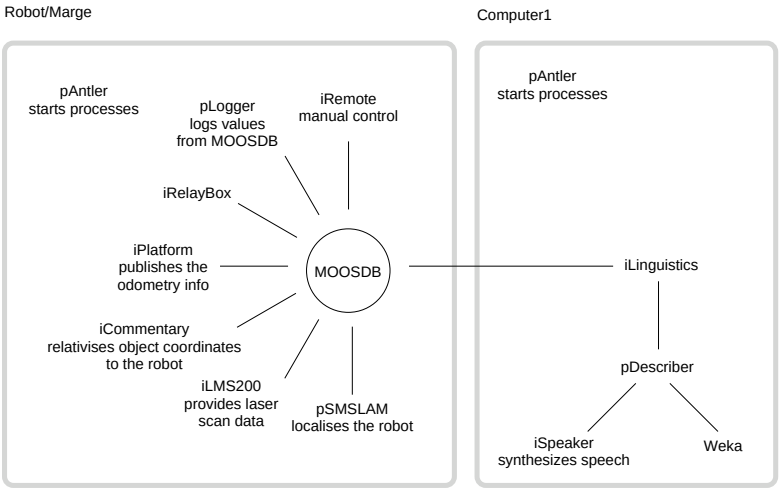


Figure 4.6: The topology of a system running *pDescriber*

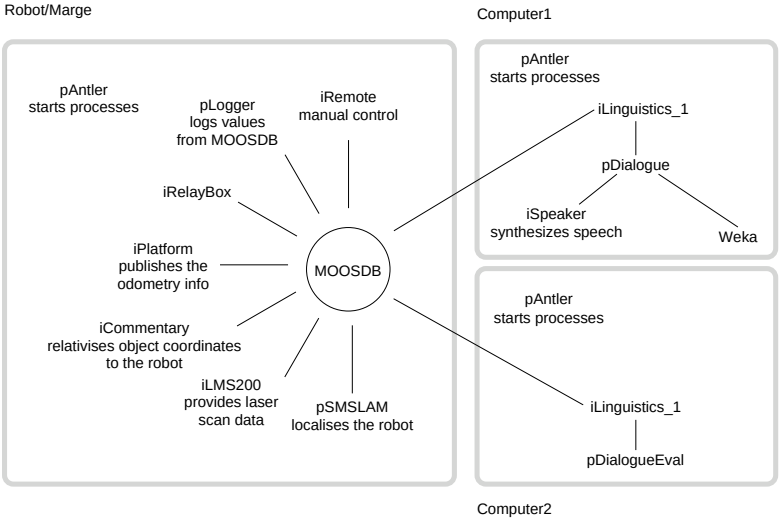


Figure 4.7: The topology of a system running *pDialogue*

Before the system can start, the above processes need configuring and the information about the environment must be collected and built. The majority of steps are

identical to the ones that were required when creating datasets for machine learning. We briefly summarise them again. First, we need to set up a new environment with new objects. Once this is accomplished a SLAM map can be built. As stated before, this is done by guiding the robot manually around the room and setting the *pSMSLAM* to the "SLAM" mode in which it builds a coherent map from its observations. When the map contains enough detail, the *pSMSLAM* can be restarted in the "localisation" mode in which the system, based on the current observations, locates the robot on the map that it previously built. Having the map and its current location, the robot also "knows" the location of all other sites on the map. The map and the location of the robot can be displayed on the screen with the *uSMSView* which we use to point to locations and ground the objects. The names and the coordinates of objects relative to the origin of the robot obtained from the *uSMSView* must be saved in the *iCommentary* section of a *.moos* file. This process is subsequently responsible for re-calculating the global coordinates of these objects to coordinates relative to the current location of the robot. If we are running an experiment in which data is to be collected then it is crucial that correct MOOS variables are being logged. This can be checked in the *pLogger* section of a *.moos* configuration file.

The linguistic applications must be configured with a few settings in their state files as shown in Figure 4.2 and 4.3. Some MOOS variables relate to a specific robot and therefore also contain its name, for example *MARGE_ODOMETRY*. In order to retrieve these variables, the linguistic application thus needs to know which robot we are using. This can be specified as *vehicle_name* in their state files and must correspond to the *VehicleName* given in the *iAGV/iPlatform* configuration block of a *.moos* file.

Equally, the value of *maximum_speed* in both state files must correspond to the value of *MaxTransSpeed* in the *iAGV/iPlatform* configuration block of a *.moos* file and the value of *max_angular_velocity* must correspond to the value of *MaxRotSpeed*. The linguistic applications must know the maximum speed in m/s and the maximum rotational speed

in rad/s that the robot is allowed to achieve. This is because the values of speed and angular velocity learned by the classifiers are normalised against these values (see Section 3.3.2, page 66). To apply an actual value in a current setting, the predicted normalised value must be "de-normalised" using the current maximum value.

The coordinates of objects included in the classifiers have also been normalised. These too must be "de-normalised" against the current maximum room size. To estimate this we use the same algorithm that we used when we created instances from the object relations dataset (see page 67). The programme *Find-Room-Size* takes a map file created by SLAM for the current room and return the maximum x and y coordinates which must be saved as *room_x* and *room_y* in the state files.

An important section of the state files is the list of classifiers that the system should use. Because we will be testing the performance of the system on a number of classifiers that come in sets such as J48 and Naive-Bayes we need an automated system with which we can manage them. We designed two programmes to accomplish this task: *Classifier Builder* and *Classifier Chooser*.

Classifier Builder runs every arff instance file (*data.arff*) found in the current folder through every Weka machine learning classifier that we specify. It follows the Weka convention that the last declared attribute is the target class. For each run it generates a set of files. The first file stores the model that Weka learned and can be re-loaded to Weka to classify new instances (*data_classifier-name.model*). The linguistic applications need to know some information about the models before they can apply them in Weka. Most importantly they need to know what are their attributes and their possible values. Weka model files are binary and reading them is not straightforward. However, as shown in Figure 3.9, page 73 all the required information is present in plain text in the header of each arff instance file. For this reason, we extract these headers and save them separately (*data.header*). The printed results from the classifier evaluation discussed in the previous chapter are also saved (*data_classifier-name.txt*). Finally, we

need the information how the set of files was created. The name of the learner, both a common name and the Java class-path name, and any options that are passed to it when the model was built must also be passed together with the model to Weka when classifying new instances. Having the names of the files in a particular configuration is also very handy. All the above information is saved to a classifier configuration file (*data_classifier-name.pl*). If a particular classifier needs to be applied, we first consult this file and refer to the supporting files when processing the data.

While *Classifier Builder* allows us to quickly create and save classifier configurations, the purpose of *Classifier Chooser* is to apply these configurations to the *pDescriber* or *pDialogue* state files automatically to reduce the possibility of an error. The programme obtains a list of required target concepts and compares it to the definitions already present in the state file. The user has the option to leave these definitions unchanged, to delete them, or to provide a new classifier. In the latter case the programme queries the user to provide a configuration file created by *Classifier Builder*. *Classifier Chooser* first checks the header file from the given configuration if the names of the required and the supplied target concepts match. If they do not, the supplied classifier configuration is rejected. This prevents the user accidentally specifying a wrong classifier for a particular target concept. We can save different versions of state files depending on which set of classifiers we would like to apply in our experiment and quickly switch them over as required.

Once the preceding options are configured the systems are ready to run. The state files of *pDescriber* and *pDialogue* also allow some other adjustments which we do not discuss here (see Figure 4.2 and 4.3). In the following sections we turn to the overview of algorithms that make each individual linguistic application and discuss their functionality.

4.4 pDescriber

4.4.1 *pDescriber* in general

pDescriber is a commentator that replicates the knowledge encapsulated in the classifiers by generating natural language descriptions. If the robot is moving, *pDescriber* describes its motion, if it is stationary, it it describes the location of objects in its environment. The descriptions it generates are full English sentences. Grammatically, they are quite simple and overall correspond to descriptions that human commentators produced when collecting datasets for machine learning: "You are going forward slowly" or "The chair is behind the table". The variation in these descriptions is quite small and is mostly restricted to the number of adverbs that we find in descriptions of motion.

There are a few general requirements according to which the descriptions should be generated. Firstly, since *pDescriber* is run in sessions at intervals that occur every second or so we must make sure that not too many descriptions are generated. This temporal bottleneck is not due to the speed at which descriptions can be generated but the speed at which they can be pronounced by the speech synthesiser. It is solved by limiting *pDescriber* to generate descriptions in every n-th session only and do nothing in others. This is achieved by keeping a record of a session count. If the current count is perfectly divisible by n, the programme should proceed with the evaluation of other predicates or it should exit otherwise (see *describe_this_session/0* in Figure 4.8). The value of n can be specified as *objects_commentary_frequency* in the state file of *pDescriber*. Experience has shown that a good value is 3. This method of restricting the number of descriptions is only used when generating descriptions of object relations.

We also do not want the system to generate the same description again and again but only say something when there is something new to describe. The problem is particular to descriptions of motion which refer to the state of the robot which may remain unchanged for a while. A repeated generation of a description "I'm moving forward"

is linguistically irrelevant. Generating a description of object relation involves an extra step of selecting a pair of objects for which the classifiers predict the relation category. With a relatively large number of objects it is unlikely that the same pair of objects is picked consecutively twice. We solve the repetition problem with motion descriptions by remembering each time the motion categories that were generated by storing them in the state file under the *said_previously* variable. On the subsequent run, we compare the stored values with the newly generated ones. If *at least one* category has a different value, the generated values are considered, turned into sentences and pronounced, otherwise they are discarded and no sentence is generated. The method also achieves that descriptions are not generated too often and hence the method described in the previous paragraph is redundant when this method is used.

Finally, the system should alternate between generating descriptions of motion and descriptions of object relations. It is most natural to describe the motion of the robot while the robot is moving and to describe the relation between the objects when the robot is stationary. Although this could be determined by examining the odometry information of the robot, we decided it is much simpler to integrate this check with the preceding one: a description of object relation is generated when the verb from the previously generated motion description is "stopped". Yet there is a price to pay for this simple design. After classifying and generating a description of object relation, we also need to classify for the motion categories and save but not pronounce them as the value of *said_previously* which will be considered on the subsequent run. Otherwise the system would never stop describing object relations. In this respect the classification of object relation categories relies on the classification of motion categories. Difficulties could arise when the latter classifiers were particularly bad and as such they would never predict the category "stopped". In practice this has never been observed and the system always smoothly and promptly changes between the two modes of description.

Figure 4.8 shows a simplified Prolog pseudo-code that outlines the main structure

of *pDescriber*.

```
p_describer :-
   said_previously(Verb,Direction,Heading,Manner),
   decide_what_to_describe(Verb,Direction,Heading,Manner).


decide_what_to_describe(stopped,_,_,_) :-
   describe_this_session,
   describe_relation(LO,REFO,Relation),
   pronounce(LO,REFO,Relation),
   ...
   describe_motion(Verb,Direction,Heading,Manner),
   remember_what_i_said(Verb,Direction,Heading,Manner).

decide_what_to_describe(OldVerb,OldDirection,OldHeading,OldManner) :-
   describe_motion(Verb,Direction,Heading,Manner),
   different_motion_descriptions(OldVerb,Verb,OldDirection,Direction, \
      OldHeading,Heading,OldManner,Manner),
   pronounce(Verb,Direction,Heading,Manner),
   ...
   remember_what_i_said(Verb,Direction,Heading,Manner).
```

Figure 4.8: An overview of *pDescriber*. *p_describer/0* starts by retrieving the values for motion categories and queries another predicate *decide_what_to_describe/4*. If *Verb* is "stopped" and this is the nth session in which a description should be generated (*describe_this_relation/0* is true), we proceed by generating a description of object relation. The generated categories are pronounced as a sentence. We also generate descriptions of motion and save them to the state file so that the verb can be checked at the beginning of the following session. If *Verb* is different from "stopped", new motion categories are generated and compared whether they differ from the previous ones. If so, they are pronounced in a sentence and remembered for the checks in the following session.

4.4.2 Describing motion

The steps required to generate a description of motion (*describe_motion/4* and *pronounce/4* in Figure 4.8) involve analysing the odometry information that we receive about the current state of the robot and sending it to a classifier that predicts a particular category, and finally using the categories to form coherent sentences.

The odometry information is processed in the same way as when instances were created for machine learning. First, the overall speed of the robot is calculated from the *vx* and *vy* components reported by the system. Its sign is determined as described in Section 2.4.2.1, page 37ff. The obtained value is normalised to the value of *maximum_speed* defined in the state file. In a similar way the angular velocity (*vh*) is nor-

malised to the value of *max_rotational_speed* and the obtained result is used as the value for the attribute *Delta-Heading*. The normalisation step is required, since there is no guarantee that the above maximum values were the same between different runs of the system.

The two attribute values for *Speed* and *Delta-Heading* are subsequently applied to the four classifiers for the concepts *Verb*, *Direction*, *Heading* and *Manner* using the *classify/5* predicate which we describe in Section 4.4.3. The predicted values are returned as the arguments of *describe_motion/4*.

Generating sentences from the predicted motion categories is straightforward (*pronounce/4* in Figure 4.8). There are only two grammar rules. If the predicted value of the the verb category is "stopped" then the sentence "I stopped" is returned. Otherwise, a sentence is concatenated according to the following pattern: "I'm *Verb Direction Heading Manner*". When creating instances we inserted a category "none" for every attribute if no word for that attribute could be found in the description. Now we must replace every predicted "none" with an empty string. Furthermore, some categories predicated by the classifiers consist of two words or more and to keep things simple for Weka when creating instances we converted spaces to underscores using our spelling correction mechanism. Here we also remove these underscores and replace them with spaces. The verb is already in the present continuous *ing*-form as collected for the dataset and thus it does not need to be changed.

4.4.3 Working with Weka classifiers

As discussed in Section 4.2 Weka expects properly formatted text files in order to classify an instance. In our code the interface to Weka classification is handled by a wrapping predicate *classify(+Target,+[(attrib,val)],-Class,-Confidence,-Actual)*. As is suggested by the notation, the predicate takes the name of the target concept attribute and a list of pairs consisting of attribute names and their values. The predicate returns the pred-

icated class, the classifier confidence and the actual value. In our case the latter is always unknown but we keep this for the sake of completeness.

The predicate starts by examining the classifier configuration for a given target concept specified in the state file, for example the value of the feature *heading* in Figure 4.2. This information contains a full Weka path of the classifier (*weka.classifiers.trees.J48*) and the names of the files containing the trained model and the header of the arff file from which the model was built. We check again whether the last attribute from the header file corresponds to the target concept to be classified. Then we create an arff file that will be passed to Weka. There is no need to write the header again: the contents of the header file can be inserted as the header. For the data section we must write the supplied values from the attribute-value pairs in the same order as defined in the header file and separate them by commas. This step also ensures that all the attributes that the classifier expects have been given as attribute-value pairs. Of course, no value is written for the last attribute which is the target class. A question mark (*?*) is inserted instead. Finally, the arff file, the file containing the classifier model, the Weka path of the classifier and the index of the target concept attribute are all passed to Weka as command line arguments as shown in Section 4.2, page 142. The Weka output is piped back to the Prolog code, it is parsed, and the predicted class, the classifier confidence and the actual value are returned.

4.4.4 Describing relations between objects

Generating descriptions of object relations is very similar to generating descriptions of motion. First, the data from MOOS must be collected and processed. Two objects must be selected and their locations are sent to a classifier to find a description of their relation. All categories are concatenated to a sentence and pronounced.

pDescriber first needs to know what objects there are that can be described. This is accomplished by parsing the "COMMENTARY_RELATIONS" MOOS variable which is published by *iCommentary* (Section 2.4.2.2, page 44). Each object is represented as a

predicate with four arguments which are the name of the object and its x, y and z coordinates relative to the current position of the robot, for example *object(desk,0.7573,0.3421,0)*. The objects coordinates are normalised to the size of the room defined in the *pDescriber*'s state file (*room_x* and *room_y* in Figure 4.2) and the z coordinate is normalised to the robot's height also defined there as *robot_z*. Currently, the height of objects is not considered as a classifier attribute. The option is preserved for the reasons of completeness since it is reported by *iCommentary* whenever a 3-dimensional laser is used. If we use a 2-dimensional laser, the reported values of the z coordinate are 0. In addition to the object definitions that can be retrieved from *iCommentary* we also create a predicate *object(robot,0,0,1)*. All normalised values are rounded to 4 decimal places. Considering that a typical distance in each dimension is 10 metres, 4 decimal places give us an accuracy between 1 mm and 1 cm.

To describe the relation between two objects we must choose two objects and then use a classifier to predict the relation between their coordinates. Currently, the system randomly chooses any two available objects and checks that they are not identical. For testing the classifiers this appears to be the most sensible choice. An improvement would be to select only objects in a certain range around the robot and thus generate descriptions that are contextually more relevant. Numeric attributes in Weka models have no limit on their minimum or maximum values. This means that any value of a numeric attribute returns some classification. If the distance between the objects is too great, then the strength of discrimination of a classifier to predict the best relation diminishes since such descriptions were rarely, if at all, encountered in the training set. It is not unproblematic to determine the optimum distance. It could be done by trial and error using some x and y distances deemed appropriate by a human operator. Even better would be to determine an average normalised distance between the objects from the descriptions in the the training set.

But there are also further factors that influence the choice of objects in a successful

description: the shape and orientation of objects if these have identifiable sides, their individual functions, and finally their relevance in the current discourse. However, these require inclusion of complex non-spatial knowledge and a detailed representation of the linguistic discourse, something that we do not yet have available. The process of interpretation or generation of spatial expressions occurs at different levels. The base level is the level of physical space. To this, other knowledge is added in the form of constraints until the expression is disambiguated just to a single pair of objects or to a single relation. Including these constraints would defeat the purpose of our experiments which is to test how well the classifiers have internalised the spatial data obtained from a mobile robot. Since additional knowledge is not negating the spatial knowledge but only constraining the selection of possible candidate expressions, it means that if our system is compared to a proper generation systems it would overgenerate descriptions rather than generate malformed ones. As discussed in Chapter 5, this is indeed the case. The evaluators often comment that they find a generated description appropriate but not the best choice given the current configuration of the scene.

Once a pair of objects is selected their x and y coordinates are passed to the *classify/5* predicate (Section 4.4.3) which interfaces with Weka and returns a relation as follows: *classify(preposition, [(lo_x,0.3555),(lo_y,0.0045),(refo_x,0.0678),(refo_y,0.5466)], Relation, Confidence,_)*.

The predicated relation and the names of the objects are concatenated into a sentence (*pronounce/3* in Figure 4.8) using the following pattern: "The *LO* is *Relation* the *REFO*". This may also generate sentences such as "The tyres is…" but this error was ignored for the moment since objects were rarely used in plural. Again the names of objects or relations may contain underscores which are replaced by spaces before the final sentence is pronounced.

4.4.5 Asking for evaluations

To evaluate the accuracy of descriptions generated by the system we designed a small evaluation component (*ask_for_judgements*/2 and /5, omitted in Figure 4.8) which queries a human user about the appropriateness of each generated category, in particular whether the generated category is a good choice to describe a particular action or relation or not. Users have the option to answer "yes" or "no". If a user answers "no", the system further queries them to provide a better description. The data is concatenated to a parseable string which contains information about the type of evaluation (motion or relation), the predicted value, the evaluator's judgement (yes or no), the evaluator's suggestion for improvement and the complete generated sentence. The resulting string is published to the MOOS database as a variable called "HUMAN_JUDGEMENT" which can be logged together with any other value to create evaluation datasets. The evaluation component can be enabled or disabled using the *ask_for_judgements* feature in the *pDescriber's* state file (Figure 4.2).

4.5 pDialogue

4.5.1 *pDialogue* in general

pDialogue uses the classifier knowledge in more sophisticated configurations than *pDescriber*. In addition to generating descriptions corresponding to states of the robot or of the environment, it is able to "understand" the linguistic material provided by a human user and respond to it: either linguistically or by performing actions. Thus, in most cases the flow of linguistic and non-linguistic information is reversed. In *pDescriber* and in data collection it was the linguistic information that reflected the state of the environment and the robot. However, in *pDialogue* the system starts with a linguistic description and brings the robot to a state referred to by the description or it has to find some configuration of environment for which the description is true. Here

are some examples of commands, questions and answers that the system is able to interpret or generate:

(1) What can you do?

– I can follow your commands to move around the room and I can answer questions about the location of objects.

(2) Go forward slowly.

Go forward right fast.

(3) Where is the table?

– The table is to the left of the chair.

Where are you?

– I'm behind the sofa.

(4) Is the table to the left of the chair?

– Yes, the table is to the left of the chair.

– No, the table is near the chair.

(5) What is to the left of the chair?

– The pillar, the tyres and the wall are to the left of the chair.

(6) What is the chair to the left of?

– The chair is to the left of the table, the desk and the wall.

pDescriber can generate descriptions from unambiguous categories of non-linguistic information straight away, but *pDialogue* must first interpret linguistic information from a human interactor and decide what response should be taken. Therefore, after starting *pDialogue* the first step is to obtain the user utterance, parse it and decide which of the above commands or questions it belongs to. Each of these is handled by a *response module* which processes the extracted linguistic information, performs any

required classifications and generates a response. The basic structure of *pDialogue* is shown as a simplified Prolog pseudo-code in Figure 4.9.

```
p_dialogue :-
   prepare_p_dialogue,
   ask_for_judgements,
   get_user_utterance(Words),
   Words \= [],
   respond_to_words(Words).

p_dialogue.


prepare_p_dialogue :-
   prepare_classifiers([verb,direction,heading,manner, \
      delta_heading,speed,preposition,lo_x,lo_y,refo_x,refo_y]),
   create_objects,
   generate_lexicon.


respond_to_words(Words) :-
  argument_parser(Words,Verb,Parses),
  choose_the_most_likely_parse(Parses,Parse),
  response_module(Verb,Parse,Words).
```

Figure 4.9: An overview of *pDialogue*. *p_dialogue/0* consists of three parts. First, we ensure that all the required information is available and that it is structured in a useful format (*prepare_p_dialogue/0*). Then we listen to, recognise and tokenise the user utterance to words (*get_user_utterance/1*). Finally, we attempt to interpret the utterance, generate an answer and pronounce it (*respond_to_words/1*).

4.5.2 Preparing *pDialogue*

Before *pDialogue* is ready it must check whether information that the system will be processing is available and that it is in the form that can be easily and efficiently accessed by its procedures. This reduces a lot of redundant work. Consider a situation where the system successfully performs a number of complex procedures but then it suddenly fails because the required classifier is missing. Equally, it is often the case that information such as classifier information is required at more than one place in the programme. Re-parsing it from the input files creates unnecessary redundancy of operations and should be avoided. In Figure 4.9 the procedures that prepare the data are grouped under the predicate *prepare_p_dialogue/0*.

The predicate *prepare_classifiers/1* prepares all the information that is required for work with the classifiers. In *pDescriber* this is done only just before the classifiers are applied (Section 4.4.3, page 151) but there this information is not required elsewhere. The predicate checks whether the names of the target concepts supplied to it in a list argument are properly associated with Weka classifiers that are defined in the *pDialogue* state file. As before, this is done by matching each target concept with the last attribute in the corresponding classifier header file. Secondly, the predicate extracts the names and the nominal values of attributes from the header files and makes sure that the order of attributes is preserved. For each classifier a predicate *classifier(Target,N,WekaClassifier,ModelFile,AttribsVals)* is asserted where *Target* is the classifier's target concept, *N* is its number of attributes, *WekaClassifier* is its full classifier class path, *ModelFile* is the name of the file with the classifier model and *AttribsVals* is a list of tuples *(attribute,[val1,val2,val3...])*. If the attribute type is numeric or string, then this structure is simply *(attribute,[numeric])* or *(attribute,[string])* respectively.

The predicate *create_objects/0* asserts *object/4* predicates containing the names and normalised coordinates of objects in the current room from the MOOS data as described for *pDescriber* in Section 4.4.4, page 152. Because users now refer to the robot as "you", the predicate *object(you,0,0,1)* rather than *object(robot,0,0,1)* is asserted.

The last predicate *generate_lexicon/0* creates a linguistic lexicon with a simple grammar that *pDialogue* can use to parse user input. Why does the lexicon have to be created rather than be pre-defined? The system can only "understand" or make sense of those lexical items which are grounded or are associated with non-linguistic information. This information changes with context, for example the names of objects that are present in the room and the range of vocabulary that the chosen classifiers can deal with. The knowledge is defined in the system but comes in different forms and from different sources: (i) from the attribute values contained in *classifier/5* predicates, (ii) from the names of the objects asserted in *object/4* predicates and (iii) from some

pre-specified lexical entries that provide grammatical glue for extracting simple dependency relations from the user input. The task of *generate_lexicon/0* is to extract this knowledge and to check its completeness.

An attribute may occur in more than one classifier either as a target concept or as a regular attribute. For example, in Table 3.4 on page 79 we can see that *Relation* is one of the target concepts predicated by the classifiers but it also occurs as a regular attribute in classifiers for *LO_x*, *LO_y*, *REFO_x* and *REFO_y*. When generating lexical entries from classifier attribute values it is important that we *only* create lexical entries for those values of a particular attribute that are defined for *all* occurrences of that attribute in the currently chosen set of classifiers. This is ensured if all classifiers are built from the same dataset. However, if classifiers from different datasets are combined, a check must be made. Our intuition is that if a value of a particular attribute cannot be used by every classifier in the currently applied set, then its reference is incomplete. The system does not have a coherent knowledge about the meaning of that value. Thus, it is better to exclude such word from the vocabulary straight away rather than finding out later that it cannot be interpreted in some cases.

We collect attribute values from which lexical entries are created by examining members of *AttribsVals* from the currently asserted *classifier/5* predicates. Thus, we may find tuples such as *(relation,[left,right,near])*, *(relation,[right,near,behind])* and *(relation,[left,right,behind])* as the members of *AttribsVals* in the classifier definitions for *Relation*, *LO_x* and *REFO_x* respectively. The set of lexical items for the category *relation* is built by finding an intersection between the lists of attribute values across all the tuples. In this case, the intersected set only contains one member, the value "right".

Different discretisations of a numeric attribute will create different nominal categories. Intersecting such sets of values will result in an empty set. We may also come across the original numeric attribute whose value will be *[numeric]* which equally gives an empty intersection. This is not a difficulty: the discretised numeric attributes do not

represent linguistic information and hence their values are never used to generate lexicon entries and can be ignored.

The collected attribute values must undergo a small morphological modification before they can be turned into lexical entries. The values of the attribute *Verb* occur in classifiers in *ing*-form since this is how they appear in descriptions from which they are extracted. In the user input that we want to parse verbs appear as imperatives and thus the attribute values must be rewritten to their basic forms. This is done by a simple verb stemmer which applies the following character replacement rules on every value of the *Verb* attribute: (i) word final "ed" → "", (ii) word final "ving" → "ve", and (iii) word final "ing" → "". Another rule is applied on the output of the above rules which removes all word final duplicated consonants ("stopping" → "stop"), except if these are "ll" as in "telling" → "tell". By no means is this treatment of English verb morphology complete, for example "storing" → "*stor". It did nonetheless correctly handle our vocabulary and it was therefore considered appropriate for the task.

The values of the *Relation* attribute may also require morphological stemming. When instances were created, they were sometimes rewritten by spelling correction rules from words such as "left" to forms such as *to_the_left_of*. These forms were intended for generating descriptions in *pDescriber* but are less useful for parsing user questions and commands. Ignoring our knowledge of language, if one had to choose a single most identifiable part of such word structures, it would be the longest sequence of characters delimited by an underscore, thus "left". If there is a tie between two items of equal length, the first candidate is taken.

The final step in creating a lexicon from the classifier data involves rewriting attribute values as lexical entries that can be used by our dependency parser. We assign every attribute, for example *Relation*, a particular lexical pattern such as *lexical_template(relation,[cat=p,sem_type=relation,arg_list=[]])*. This is the only knowledge that is added manually when creating the lexicon. The intersected values of each attribute,

for example "right", are then rewritten as predicates *lexical_entry([form=right,stem=right|Features])* where *Features* corresponds to the second argument of *lexical_template/2*, thus the list *[cat=p,sem_type=relation,arg_list=[]]*.

There are two cases when no *lexical_entry/1* is asserted. The first case is when the attribute value is "none". This attribute value was used when no word of that category was found in the corpus description (Section 3.3.3, page 67ff.). No lexical entry is also asserted if such entry already exists. This should not happen according to the preceding processes. The constraint ensures the consistency of the lexicon as some lexical entries may have been manually added to the lexicon for testing purposes.

Lexical entries for words referring to objects are asserted in the same way except that the names of the objects are taken from the *objects/4* predicates rather than from attribute values stored in *classifier/5* predicates. Classifiers contain no attributes with object names. The procedure creates predicates such as *lexical_entry([form=chair,stem=chair, cat=n,sem_type=object,arg_list=[]])*. Finally, we also assert approximately 15 default lexical entries which include question words ("what" and "where"), prepositions ("to", "of", "by" and "from") and sentential predicates ("is" and "are") to which we return in the following section. Once this is done, the lexicon is complete and ready to parse user input.

4.5.3 Parsing user linguistic input

pDialogue can only respond to users successfully if it is able to resolve the content of their questions or commands unambiguously. For representing semantics of sentences a state of the art system could be used such as (Curran et al., 2007). However, since the sentences are relatively simple and can be categorised into a small number of categories, this was deemed unnecessary. In Section 4.5.1, page 156 we have identified one command (Example 2) and five question types (Examples 1 and 3–6). The task is similar to the task in which we extracted words from the corpus of descriptions to create attribute values for machine learning described in Section 3.3.3. It is slightly more

complicated. When extracting attribute values we knew what kind of descriptions are found in each corpus that we processed. Therefore, we also knew what predicates, either verbs or object relations, and their arguments we should find in these utterances as discussed on page 69. However, in *pDialogue* we may encounter any user utterance and therefore must also recognise its type.

We assign each type of sentence (2–6) from page 156 a main predicate which identifies it as shown in (8–12). Lexical entries such as (8) are created automatically from the values of the *Verb* classifier attribute as discussed on page 158, whereas entries in (9–12) are added manually. Having the verb as the main predicate rather than separate main predicates for verbs and object relations as in Section 3.3.3 allows us to unify the argument extraction procedure. It also minimises the size of the lexicon. Rather than creating lexical items for every description of object relation three times to cover the sentences in (10–12), we only need to create lexical entries for the verbs "is" and "are" which gives us only six different entries. The ambiguity between these entries is resolved by our argument matching mechanism. Overall, the lexical entries are very similar to those in Section 3.3.3.

(7) What can you do?

Sentences of this type are handled by *word_matcher/2* described in Section 4.5.4.

(8) Go forward slowly.

lexical_entry([form=going,stem=go,cat=v,sem_type=movement,
arg_list=[direction=0,heading=0,manner=0]]).

(9) Where is the table?

lexical_entry([form=is,stem=is,cat=v,sem_type=locating_object,
arg_list=[q_where=1,object=1]]).

(10) Is the table to the left of the chair?

lexical_entry([form=is,stem=is,cat=v,sem_type=object_description,

arg_list=[object=1,relation=1,object=1]]).

(11) What is to the left of the chair?

lexical_entry([form=is,stem=is,cat=v,sem_type=finding_object,

arg_list=[q_what=1,relation=1,object=1]]).

(12) What is the chair to the left of?

lexical_entry([form=is,stem=is,cat=v,sem_type=referencing_object,

arg_list=[q_what=1,object=1,relation=1,preposition=1]]).

The extraction of the verbal predicate and its arguments from mono-clausal sentences is handled by *argument_parser/3* which is outlined in Figure 4.10. The difference between this argument parser and the one described in Section 3.3.3 is that this argument parser does not return a single parse but a set of parses, some of which may be incomplete. Later a heuristic is applied on this set of parses to choose the best candidate parse (*choose_the_most_likely_parse/2*) which is subsequently supplied to the associated response module (*response_module/3*) as outlined in Figure 4.9. It may also happen that no parse is returned for sentences such as (7). However, in order to make *pDialogue* cooperative and responsive, such cases must also be covered.

```
% argument_parser(+Words,-Verb,-Parses)

  argument_parser(Words,Verb,Parses) :-
     find_verb(Words,Verb),
     find_verb_interpretations(Verb,TypeArgs),
     process_interpretations(Words,TypeArgs,TypeWords),
     reject_incomplete_interpretations(TypeWords,Parses).
```

Figure 4.10: An outline of *argument_parser/3*.

Let us assume that a user asks the robot a question "Where is the chair?". The argument parser first identifies the verb from its list of tokenised words (*find_verb/2* in Figure 4.10). The verb is the first *Word* on the list of the utterance words that unifies with the predicate *lexical_entry([form=_,stem=Word,cat=v|_])*. This identifies the word

"is". We use the stem feature rather than the form feature to match the word because commands contain verbs in their "bare" form, for example "go" rather "going". If no verb covered by our grammar is found, then "none" is returned.

Given our grammar it is likely that the verb that was extracted will match more than one lexical entry each of which has a unique *sem_type* specification as shown in (8–12). In the following step (*find_verb_interpretations/2*) we compile a list of semantic types *Types* that are associated in the lexical specifications with a particular verbal stem. Since each semantic type has a unique argument specification we also extract this which simplifies the following step. The returned list *Types* contains tuples of the form *(Type,Args)*. In the case of the verb "is" the list in Figure 4.11 is returned.

```
[ (locating_object,[q_where=1,object=1]),
  (object_description,[object=1,relation=1,object=1]),
  (finding_object,[q_what=1,relation=1,object=1]),
  (referencing_object,[q_what=1,object=1,relation=1,preposition=1])
]
```

Figure 4.11: Possible interpretations of the verb "is"

Having a list of possible sentence interpretations defined as predicate-argument dependencies we must now check which of these interpretations fits the given sentence best (*process_interpretations/2*). This is done by checking whether the sentence contains words that match the arguments in *Args* of each *(Type,Args)* tuple in the same way as described in Section 3.3.3, page 67. Because the same mechanism is used to parse both descriptions of motion and descriptions of object relations, a distinction between obligatory arguments such as "chair" in (13a) and non-obligatory arguments such as "fast" in (13b) must be also taken into account (see page 71).

(13) a. The desk is to the left of the chair.

b. You're going forward fast.

The two kinds of arguments have different syntactic and semantic properties. Obligatory arguments cannot be left out (14a) whereas non-obligatory can be (14b). Another property of obligatory arguments in English and in other configurational languages is that they occur in fixed positions in a sentence structure. (15a) is semantically different from (15b). Sentences may contain none or potentially an unlimited number of non-obligatory arguments of the same semantic type (16a). The only requirement is that they are semantically compatible (16b). The descriptions within categories *Direction*, *Heading* and *Manner* all refer to semantically exclusive concepts ("fast" $\neq$ "moderately") which means that they cannot be used at the same time.

(14) a. *The desk is to the left of the.

b. You're going forward.

(15) a. The chair is to the left of the desk.

b. The desk is to the left of the chair.

(16) a. Peter read a letter in the evening at eight o'clock.

b. *Peter read a letter in the evening at 3am.

The types of arguments are encoded in the lexical specifications of predicates in (8–12) with the flags 1 and 0 where 1 means that the argument is obligatory and 0 that it is not, for example *[q_where=1,object=1]*.

The predicate *process_interpretations/2* considers every tuple *(VType,Args)* in *[(VType,Args)]* and every *Arg* in *Args* whether the *Arg* can be matched with the *sem_type* feature of any of the words in the utterance.[6] If it can, that word is returned. If the list of words is exhausted and no word is matched, then "missing" is returned if the argument is obligatory and "none" if the argument is non-obligatory. The procedure continues to match words for all remaining *Args* of all remaining tuples.

[6]Since there is no parallel check on the category feature (*cat*), the method implies that semantic types such as *object* are not linked to lexical categories such as nouns which linguistically is an oversimplification. However, for our purposes encoding this feature of grammar is unnecessary.

The fixed order of obligatory arguments is taken into account by limiting the list of words that is supplied to the search. When matching the first *Arg* of each *Args*, the procedure starts with a complete list of tokenised words. It creates a list of words *ArgWords* in an identical order to the order of arguments in *Args* to which they are matched. If an obligatory argument is matched to a word, then the search of the second *Arg* in *Args* is only supplied with a list of words that follow the matched word on the original list of words. On the other hand, if a non-obligatory argument is matched to a word, the search receives the original list of words excluding the word that was matched. This ensures that non-obligatory arguments can occur on the list of words in any order and that the same word is not associated with more than one argument. If more than one word of the required semantic type is present, the first word of that type is taken and all other words are ignored. If the word for a particular argument is "missing" or "none", then an unchanged list of words is used for the next iteration of the search.

Our treatment of argument search has implications on the order of obligatory and non-obligatory arguments in *Args*. In most cases, but also depending on the rule that we want to encode, non-obligatory arguments should be declared before obligatory ones. For example, *went:[manner=0,agent=1,location=1]* would parse sentences "Ann went to the house", "Slowly, Ann went to the house", "Ann went slowly to the house" and "Ann went to the house slowly". However, a specification such as *went:[agent=1,location=1,manner=0]* would only parse the last sentence.

To summarise, the predicate *process_interpretations/2* rewrites a list possible interpretations of a sentence encoded as a list of tuples in Figure 4.11 to a list of matched interpretations also encoded as tuples shown in Figure 4.12. Our task is to pick a single tuple and reject all others. Tuples where "missing" is a member of the matched words can be rejected straight away since this indicates that an obligatory argument could not be matched (*reject_incomplete_interpretations/2*). In the example in Figure 4.12 this selects

(locating_object,[where,chair]) as the final candidate. If a sentence is matched to rules that contain non-obligatory arguments, the tuples may contain one or two occurrences of "none" depending on how well each rule matches the sentence. In this case, we prefer a tuple with the fewest occurrences of "none", or in other words, a tuple with the greatest number of matched arguments. This is the most meaningful interpretation of such utterance since it contains the greatest amount of semantic information. If there is a tie between two candidates, for example if the grammar writer introduced ambiguity, we simply take the first one. When designing our grammar we made sure that this was never the case. This type of filtering is performed on the list of interpretations after they are returned by *argument_parser/3* (*choose_the_most_likely_parse/2* in Figure 4.9).

```
[ (locating\_object,[where,chair]),
  (object\_description,[chair,missing,missing]),
  (finding\_object,[missing,missing,chair]),
  (referencing\_object,[missing,chair,missing,missing])
]
```

Figure 4.12: Matched interpretations of the verb "is"

There are two situations where the argument parser may not extract any semantic information from the supplied words. This may arise if no verb that can be matched to the given lexicon is found on the list of words. Secondly, this may arise if all parses are incomplete and are thus rejected. In both cases the argument parser returns "none" for the verb and an empty list [] instead of a list of matched tuples.

4.5.4 Matching user linguistic input to word patterns

The system only contains a fragment of English grammar that is sufficient to parse questions and commands which can be answered using the knowledge from the MOOS and the classifiers. One of our aims is that our system should be linguistically co-operative and therefore must handle a few other situations. A user may not know what the system's competence is and thus may pose questions that are not handled

by our response modules. Sometimes their input may contain words which were misrecognised by the speech recogniser. They may request for clarification and help on the system's performance. All such cases should be handled in a linguistically cooperative way.

These situations could be covered by *argument_parser/3*. However, in such case the set of grammar rules would have to be expanded substantially. Furthermore, there is also a difference in the purpose of both tasks. *argument_parser/3* strives to provide a unique unambiguous parse. The accuracy of this parse is crucial since it is expensive for the system to produce a wrong response. This is because irrelevant processes would be performed which would give us bad evaluation results or even cause damage to the equipment. On the other hand, the accuracy is less important for other linguistic situations. We want to be able to issue a response even if the user input can only be partially matched. In fact, we may rely on similar meanings of a group sentences and respond to them in the same way. Answering something is better than not answering at all.

To this end we developed a simple pattern matching dialogue interface called *word_matcher/2*. This provides a robust handling of variable user linguistic input. As the name implies, the word matcher takes a list of tokenised words and compares them with words defined in pattern predicates. The pattern with the highest numbers of words in common with the user input wins and its associated response sentence is returned as the robot's reply to the user.

Patterns are defined as three-place predicates *pattern(?Id,?[Keywords],?Reply)* where *Id* is a unique pattern identifier, *KeyWords* is a list of words that define the pattern and should be matched, and *Reply* is the reply that the system gives in case a match with the user input is made. An important feature of our approach is that *KeyWords* does not contain complete sentences but a bag of words all of which are related to a certain reply as shown in Figure 4.13.

```
pattern(2,[help,what,you,can,do,commands],'I can move around the room
   or tell you where the objects are. Just tell me what I should do.').
```

Figure 4.13: A linguistic pattern

The choice of *KeyWords* is important for the accuracy of the system. Frequently used words such as auxiliary verbs and prepositions are excluded. It is to expect that by increasing the size of the database of patterns the accuracy of choosing the correct pattern will be reduced since it is more likely that the same word will occur in more than one pattern. To counter this difficulty we could extend our count of matches with a measure of *specificity* which is known in Information Retrieval as *inverse document frequency* (IDF) (Spärk Jones, 1972). IDF is defined as a logarithm of the ratio of the number of documents in corpus over the number of documents containing at least one occurrence of the word. It therefore tells us how specific or general the word is in the collection of documents or in our case in a collection of pattern definitions.

$$Sp_i = log\frac{|P|}{|\{p_j : w_i \in p_j\}|} \tag{17}$$

In Equation 17 $|P|$ is the total number of patterns in our database and $|\{p_j : w_i \in p_j\}|$ is the number of patterns that the word w_i appears in ($n_{i,j} \neq 0$). Instead of counting each match as 1, we could weight it by Sp_i and sum the weights. A high weight is obtained if a particular term occurs rarely in patterns and a small one if the term occurs frequently.

An alternative measure is the Dice's coefficient which also comes from Information Retrieval (van Rijsbergen, 1979, page 39). This is a measure of similarity which is defined as the ratio between the number of items of two sets as shown in Equation 18.

$$s_{ij} = \frac{2|I \cap J|}{|I| + |J|} \tag{18}$$

$|I \cap J|$ stands for the number of terms that the sets have in common. S_{ij} ranges from 0 to 1. $s_{ij} = 1$ indicates that the two sets of terms are identical and $s_{ij} = 0$ that they are completely different. However, our pattern definitions do not contain utterances that could be matched to user utterances in this way. Examining the pattern in Figure 4.13 it can be seen that the list of words would match at least three different utterances such as "Please, help me!", "What can you do?" or "What commands do you understand?" as well as any permutation of these. When using the Dice's coefficient we would have to define one pattern per each user utterance. On the other hand, our alternative measure allows us to be more expressive with our definitions of patterns and cover a broader range of user input.

word_matcher/2 takes a list of tokenised words *+[Words]* and returns an atom or a string as *Reply*. First it examines the words and builds a scoreboard which contains a list of items such as *[pattern1=5,pattern3=1]* asserted in a predicate *scoreboard/1*. The items tell us the number of words in the intersection between the supplied *Words* and a particular pattern definition. The scoreboard is initially empty and is incrementally built as follows. For every word in *Words* a list of pattern IDs is created which contain that word in their definition, for example the word "how" is found in patterns *[3,6,9,10]*. If this list is empty, we can proceed with another word. Otherwise, the items from the list must be added to the scoreboard. Every pattern ID on the list of matched patterns IDs increments the score of the same pattern on the scoreboard list by one. If the pattern is not present on the scoreboard yet, it is added with a score of 1. Hence, with the above list of patterns IDs a scoreboard such as *[1=2,3=4,4=1,6=2,8=1,9=1]* is updated to *[1=2,3=5,4=1,6=3,8=1,9=2,10=1]*.

Once we evaluated all words in the user's input we need to choose the pattern with the highest score. If the scoreboard is empty, the pattern ID *none* is returned which is reserved for a pattern with a reply such as "I'm sorry, I don't understand". If the highest score on the scoreboard is shared by more than one pattern, the selection simply

prefers the first pattern that is encountered with that highest score and rejects all the rest. The order of items on the scoreboard reflects the order of words in *Words*. The procedure starts with the first word and updates the scoreboard with scores. New items are added to the scoreboard as heads of the list which means that a preferential bias is given to words closer to the end of the sentence. This does make sense linguistically since new information in English sentences is contained in the second part of clauses.

The mechanism is very simple but works surprisingly well in practice. It could be extended to provide non-linguistic replies or we could use pre-defined word templates to extract and remember certain types of information from the user input such as their name and reuse this information as variables in the patterns replied ("Pleased to meet you, John.").

4.5.5 Responding to user

In the preceding two sections we discussed how user linguistic input is interpreted. Now we turn our discussion to how the system is able to generate a response. In general, all user linguistic input handled by our system are requests for information. To provide a response the system must combine the information from a user's request, the information that it has about the state of the robot and the environment, and the knowledge embodied in the classifiers. This is a typical task of dialogue systems or embodied conversation agents such as CHAT (Weng et al., 2007) and COMPANIONS (Wilks, 2006) in which the steps required to generate a response are automatically derived from multiple sources by some reasoning mechanism. *pDialogue* does not implement automatic reasoning. Instead the steps required to respond to the user are coded by hand and grouped together as sets of operations called *response modules*.

There are six response modules, each handling one group of sentences (1–6) on page 156. We refer to the response modules by the semantic type of the verb that is identified in the input sentences by our grammar rules as shown in (7–12) on page 162. Each response module is represented as a predicate *response_module/3* which takes three

arguments: *Verb*, *Parse* and *Words* (Figure 4.9, page 157). Note that *Parse* is a tuple consisting of a semantic type of verb and a list of the matched argument words, for example *(locating_object,[where,chair])*. *Words* is an unparsed list of words in the original utterance and is only required for the module containing *word_matcher/2*.

4.5.5.1 Generating motion

The *movement* response module generates robotic motion following user's linguistic commands. It takes takes words for *Verb* ("going") and *Direction*, *Heading* and *Manner* from the supplied parse, for example *(movement,[none,left,none])*, and applies them to the classifiers for *Delta-Heading* and *Speed*. The predicted categories are subsequently rewritten to commands that bring the robot to the state referred to in the description.

Verbs must be passed to the classifiers in their *ing*-forms as this is how they have been learned. *pDialogue* uses the same interface to Weka classifiers as *pDescriber* called *classify/5* which we discussed in Section 4.4.3, page 151ff. The classifiers predict *Delta-Heading* and *Speed* as discretised nominal classes, for example "0.0914-0.2742". These classes represent intervals of numeric values, in this case $0.0914 < x \leq 0.2742$. By convention accepted during the discretisation step this interval is half-closed $(Min, Max]$. The minimum and maximum values can be extracted from the interval label. Then a random number is picked from the interval such that it satisfies the $(Min, Max]$ condition.

Having the values of *Delta-Heading* and *Speed* is not enough to generate motion. The values describe a robot's state but we require instructions that bring the robot to this state. Fortunately, *iAGV/iPlatform* already includes the required mechanisms. It accepts commands to generate motion in the form of two MOOS variables "DESIRED_RUDDER" and "DESIRED_THRUST" which range from 0 to 100. These are also generated by *iRemote* when the robot is controlled through a computer keyboard. The desired rudder and the desired thrust are the percentages of the *MaxRotSpeed* and *MaxTransSpeed* variables of *iAGV/iPlatform* defined in the *.moos* file. The values are avail-

able to *pDialogue* through its own variables *max_angular_velocity* and *maximum_speed* (Figure 4.3). The values of *Delta-Heading* and *Speed* were normalised to *MaxRotSpeed* and *MaxTransSpeed* of the configuration under which they were collected. Therefore, the predicted *Delta-Heading* and *Speed* values can be applied to *iAGV/iPlatform* as "DESIRED_RUDDER" and "DESIRED_THRUST" with minimal adjustments only: they must be expressed in the range between 0 and 100 rather than 0 and 1 and the sign of *Delta-Heading* must be reversed to measure rudder rather than yaw.

Generating motion from classifier data is risky because it is likely that wrong motion will be generated which may damage the equipment. For this reason we include a safety mechanism which asks the user whether the generated values of "DESIRED_RUDDER" and "DESIRED_THRUST" should be sent to MOOS or not. This safety mechanism can be switched on or off using the *must_confirm_motion* feature in the *pDialogue* state file (Figure 4.3). If the values are submitted, the motion continues for the number of seconds specified in the *iAGV/iPlatform* configuration block of the *.moos* file. It changes if the user adjusts the values of rudder and thrust through *iRemote* or if they issue a new linguistic command.

In addition to motion the response module also generates a linguistic description which serves as an acknowledgement that the user input was interpreted properly and provides another safety mechanism. If a user issues a command such as "Go forward slowly" and it is parsed correctly, then the robot confirms it with a statement "I'm going forward slowly". The mechanism to generate these sentences has been borrowed from *pDescriber*.

4.5.5.2 Locating objects

The *locating_object* module provides answers to questions such as "Where is the table?" and "Where are you?". The purpose of these questions is to obtain a description of the location of an object relative to the location of another object, the reference object,

which is known. As in *pDescriber* the answer is generated by randomly selecting another object in the room to serve as the REFO and ensuring that the objects are not the same. It will happen inevitably that some descriptions will be judged less appropriate because of the random selection of the REFO (for a discussion see Section 4.4.4).

When the two objects are known, their x and y coordinates are extracted and are sent as the values of the *LO_x*, *LO_y*, *REFO_x* and *REFO_y* attributes to the *classify/5* predicate which applies them to a Weka classifier and returns a relation as shown on page 154. The relation, the LO and the REFO are concatenated to a sentence which is pronounced.

4.5.5.3 Confirming an object description

If both the robot and the user are familiar with the scene, a question such as "Is the table to the left of the chair?" queries the robot whether it agrees with a given relation to describe a particular pair of objects. The user and the robot may choose different relations to describe a particular set of objects and the robot's reply indicates agreement or disagreement: "Yes, the table is to the left of the chair" or "No, the table is near the chair". Note that the user can also deliberately form the question so that the expected answer is "no".

The steps involved in generating an answer in the *object_description* response module are very similar to the *locating_object* module. Here both objects are already known and their coordinates can be sent to *classify/5* straight away to predict the relation. If the predicted relation matches the relation from the question, the generated answer is "yes" otherwise it is "no". In the sentence that follows the predicted relation is used in both cases.

In a situation where the user cannot see the scene the intention of the question would be to help them build a mental image of the scene. In this case the robot would be more cooperative if it preserved the LO and the relation and reported the applicable REFO: "No, the table is to the left of the sofa". We do not concentrate on this scenario.

4.5.5.4 Finding objects

The question "What is to the left of the chair?" expects an answer containing a set of LOs for which it holds that they are close to the REFO "the chair". The module *finding_object* proceeds by taking the relation "left" and the x and y coordinates of the REFO and applies them to two classifications to find the nominal classes defining the x and y ranges of the potential LOs: *classify(lo_x, [(preposition,left),(refo_x,0.0358),(refo_y,0.2581)], LO_x,_,_)* and *classify(lo_y, [(preposition,left),(refo_x,0.0358),(refo_y,0.2581)], LO_y,_,_)*.

The minimum and maximum values are extracted from the predicted nominal classes to give two half-closed intervals $(X_{min}, X_{max}]$ and $(Y_{min}, Y_{max}]$. In the following step the *object/4* predicates are examined. An object is included on the list of the returned LOs if it lies in the square region defined by the two intervals. Its x and y coordinates must satisfy the conjunction of the following constraints: $x > X_{min} \wedge x \leq X_{max} \wedge y > Y_{min} \wedge y \leq Y_{max}$.

Forming sentence descriptions now requires a few extra steps rather than just concatenating words (Section 4.4.4). Sentential subjects and objects may contain a list of items that must be expressed as a conjunction. Although in this response module we only need to generate descriptions with conjoined subjects we also consider generating descriptions with conjoined objects which are required in the *referencing_object* module. If the subject is a conjunction of names of physical objects, the following verb should appear in plural (19b) rather than singular (19a). Subjects and objects differ in pronoun forms. The classifiers refer to the robot with a pronoun "you" which must be rewritten either as "I" (20b) or "me" (20a) depending on the argument that they represent. If the list with subject or object names is empty, it must be rewritten as "nothing" as in (21a) and 21b).

(19) a. The chair is...

b. The chair, the table and the sofa are...

(20) a. The sofa is to the left of me.

b. I'm to the left of the sofa.

(21) a. Nothing is to the left of the chair.

b. The sofa is to the left of nothing.

Items on the list must be added a definite article "the" unless they are a pronoun or "nothing". Lists of two items and more must be separated by commas and the penultimate and the ultimate item must be separated by "and". If the subject is "I", the verb must be "am". If the subject is a single item ending in "-s" or a conjunction of items, the verb is "are". Otherwise, the verb is "is". Any underscores used in the relation names must be replaced with spaces. Then, the subject, the verb, the relation and the object can be concatenated to a sentence and pronounced.

4.5.5.5 Referencing an object

The last response module *referencing_object* handles questions such as "What is the chair to the left of?". These query for a set of objects that can be used in a description as reference points (REFOs) for the LO given in the question, "the chair". The steps required to answer these questions are almost identical to the steps for the previous question type. The difference is that here we are given the LO and the relation and the classifiers must predict a square region containing the potential REFOs rather than LOs: *classify(refo_x, [(preposition,left),(lo_x,0.2386), (lo_y,0.0571)], REFO_x,_,_)* and *classify(refo_y, [(preposition,left),(lo_x,0.2386),(lo_y,0.0571)], REFO_y,_,_)*.

Following the classification, objects are extracted from the obtained intervals and description sentences are formed just as for the previous type of question.

4.5.6 Asking for evaluations

Collecting human evaluations in *pDialogue* is also more complicated compared to *pDescriber*. This is related to the fact that the system is run in sessions (Section 4.2). The

main difficulty lies in the fact that motion cannot be evaluated in the same session in which it is generated but only in the following session. The generated response must be sent to MOOS first but this can only be done when *pDialogue* exits. When the response reaches the MOOS database, its values are picked up by the relevant component such as *iAGV/iPlatform* which generates the requested motion which the user can observe and evaluate. A new session of *pDialogue* is started and the user can now enter their scores. The challenge is to match the information from the robot's response with the scores of the evaluator, a similar problem we faced when matching information for instances for machine learning.

Each response module submits a list of values used in that module to the MOOS database together with the name of the response module, the sentence with which the system replied and the current session number of *pDialogue*. The latter serves as an identifier of the entry in the MOOS log file. The variables are concatenated to an atom with separator characters which help to separate their values later. The atom is submitted to MOOS database as the value of the "CLASSIFIER_ATTRIBUTES" variable. Before completing, the procedure also changes the values of two variables in the *pDialogue* state file: *evaluate_response* is set to "yes" and *response_in_session* is set to the current session number.

When *pDialogue* is run in a new session, the predicate *ask_for_judgements/0* is one of the first predicates to be evaluated even before new linguistic input is parsed and interpreted (Figure 4.9). If *evaluate_response* is set to "yes", there is a pending evaluation of a *pDialogue*'s response from one of the previous sessions. If not, the evaluation step can be skipped and *pDialogue* can proceed. The evaluation component can be disabled by setting the value of *ask_for_judgements*, another state file feature, to "no".

The mechanism that collects scores from a human evaluator can be set to accept binary scores ("yes" or "no") to answer questions such as "Was this a good answer/motion?" or graded scores on a 5-level scale, where 1 is inappropriate, 3 is acceptable

and 5 is perfect, to answer questions such "How good was the answer/motion?". The type of scoring can be selected by adjusting the value of the *binary_judgements* feature in the state file. Evaluators enter their judgements on a computer keyboard. In addition to the score, they can also enter a short comment of their choice in the form of free text.

The value of *response_in_session* is read from the state file. Normally this corresponds to the number of the previous session in which *pDialogue* generated a response. The session number, the score provided by the evaluator and the comment description are all concatenated to a structured atom and submitted to the MOOS database as the value of the "HUMAN_JUDGEMENT" variable. When log files are processed, the pairs of entries "CLASSIFIER_ATTRIBUTES" and "HUMAN_JUDGEMENT" can be matched by the session number and the consistency of the evaluation data is ensured.

Finally, before completing the evaluation module, *evaluate_response* is set to "no" and *response_in_session* is set to -1 which indicates that on the next run of *pDialogue* nothing needs to be evaluated, unless of course a new response is subsequently generated in the same session in which case the variables are adjusted again. Both features must be reset to these values in session 0 when starting up to clear any settings from the previous run of the system (Figure 4.4).

4.6 Conclusion

In this chapter we described *pDescriber* and *pDialogue*, our two systems that use the robot's knowledge about itself and the environment, the classifiers and some linguistic knowledge to interact with human observers. When building these systems we had two objectives in mind. The systems should have a parallel structure which allows us to compare them. Secondly, the systems should ensure a flow of information. They must reuse the information that they already have and minimise redundant data as much as possible. Most lexical categories in our grammar thus only contain words that were created from the attribute values of classifiers. This reduces the need of manual

data input and the occurrence of errors as the consistency of the system's knowledge is ensured, no matter with which set of classifiers or in which environment the system is used.

The systems contain some elements of embodied conversational agents. Building a complete state of the art conversational agent goes beyond the scope of this work but is certainly a very compelling topic for future research. The systems have been built with this in mind so extensions could be implemented without having to change much of the existent base. Currently, we implemented only those features that are required for interaction with human evaluators and collecting their judgements in the experiments that we describe in the following chapter. We turn to these experiments now.

Chapter 5

Evaluation by humans

5.1 Introduction

The evaluation of machine learning classifiers in Chapter 3 tested the degree to which they fit the conceptualisations of spatial and motion expressions given a particular environment and a particular kind of information available. It may tell us quantitatively how many errors the classifiers are making and on which target class but it does not tell us why the errors are made. Observation of the performance of these classifiers in real scenes will help us to find the answers to this question. Furthermore, the training datasets may have included systematic errors that have been incorporated in the learned models and are thus predicted as perfect classifications during classifier evaluation. Similarly, the datasets may have systematically excluded certain kinds of observations which make the system either over-generate or under-generate in real scenes. In the first case descriptions referring to certain configurations are generalised to all configurations, and in the second descriptions of certain configurations are never generated. Finally, the evaluation of *pDescriber* and *pDialogue* by humans will tell us how well the models extend when some conditions, such as environment, human observers and configurations of scenes are varied and whether adjustments such as normalisation were appropriately chosen. The application of the learned classifiers in real environment is the ultimate test for the success of learning.

5.2 Evaluation of *pDescriber*

5.2.1 Experiment design

For the experiment a group of five subjects identified as *a*, *b*, *c*, *d* and *e* were chosen from a set of volunteers. Three of them were native and two were non-native but fluent speakers of English. Three subjects already participated in the original data collection experiments and two of them were new to the experiment.

Before starting the experiment a room with objects was set up as in the data collection stage (Section 3.2.2). The majority of objects were identical but their placement was different. A new SLAM map of the environment was built (Section 2.4.2.2). The maximum distances in the x and y dimensions were found from the map and their values were added to the configuration of *pDescriber*. Similarly, we also defined the maximum speed of the vehicle as 0.6 m/s and the maximum angular velocity as 1 rad/s which we recorded both with *iAGV/iPlatform* and *pDescriber* (Section 4.3). *pDescriber* normalises all observed values against these maximum values (Section 3.3.2).

Each subject was familiarised with the scene, the names of the objects and the types of motion that the robot can perform. The subjects were reminded that they should be careful to evaluate object descriptions from the perspective of the robot rather than their own perspective or the perspective of the reference object. Examples of descriptions made from each perspective were given to them. Once the data collection has started, the operator guided the robot around the room. If the vehicle was moving, *pDescriber* produced descriptions of motion, and when it was stationary, it generated descriptions of object relations. The descriptions were pronounced to the evaluator through a speech synthesiser which attempted to replicate the naturalness of the situation. After each description was made, the evaluator, sitting at a computer terminal, was queried whether the categories in a description were good or bad given the current state of the robot and the environment. If the description was judged unsuitable,

the evaluator also had a chance to provide a more suitable description. This gave us simple binary data.

In the case of motion descriptions the evaluators were queried to judge the suitability of the verb and direction, heading and manner adverbs, thus the linguistic categories for which we built the classifiers. In the case of descriptions of object relations there was only one category to evaluate, namely the relation. The evaluators tended to give additional qualitative judgements about the system performance which were noted by the operator.

All classifiers used in the evaluation were created with the J48 learner. During the evaluation we changed the sets of classifiers between those based on the *Simple* and *All* datasets (Section 3.2.1). The motion classifiers were built from a dataset created without time-shifting in case of *Simple* and with time-shifting in case of *All*. We refer to the configurations of *pDescriber* relative to the classifier set that they are using as *J48-Simple* and *J48-All*. Each experimental session took approximately one hour. The period during which each type of description was produced or when a particular set of classifiers was applied was controlled only approximately and hence the number of descriptions of each type may vary.

5.2.2 Evaluator agreement

Before comparing the performance of the classifiers on a test set and the performance of the "live" system against human evaluators it is important to establish if the latter are agreeing in their judgements. The agreement between human evaluators tells us whether they represent a single body for comparison. If the evaluators agree it means that the system has not been tuned to particular describers who participated in the collection of the corpora for machine learning but it has captured generalisations that are applicable elsewhere. A high disagreement would tell us how difficult is for the classifiers to learn a particular concept.

The agreement between evaluators can be measured using a few different measures. In Section 3.5.3 we discussed the κ coefficient which is commonly used in coding tasks. However, our experiment is not designed as a coding task but as a measurement of system performance. To calculate the κ coefficient we need a known list of items that each evaluator considered. In our experiment the set of evaluation items was different for each evaluator depending on what descriptions the robot produced.[1] This means that we can only calculate the κ coefficient to measure the agreement between an individual evaluator and the system, but not between individual evaluators.

Inter-evaluator agreement thus cannot be estimated directly on the individual items. We can nonetheless estimate whether there is some consistent behaviour between our evaluators. If there is, this would be an indication that some direct agreement would be found between them as well. The evaluators evaluated the suitability of a closed set of words produced by the system to describe the current state of the robot and the environment. We can expect that the agreement of a single evaluator will not be the same on all the words: some words are more difficult to learn than others given the chosen methods and parameters. If this is so, the difference in the ratings for words should be consistent across evaluators. Statistical correlation can be used to determine how consistent these differences are.

Table 5.1 represents the evaluated accuracies of the system per individual words using the classifiers based on the *Simple* dataset and Table 5.2 shows them for the classifiers based on the *All dataset*. By accuracy we mean the proportion of cases where the evaluators *a*, *b*, *c*, *d* and *e* agreed with the system on the suitability of a generated word relative to the total number of generations of that word in each evaluator's session. Each numeric column shows the accuracies judged by one evaluator. The last column *All* gives us the accuracies per word when all evaluators are considered as a whole. Note that this is not simply the mean of the accuracies per individual evaluators.

[1] In the evaluation of *pDialogue* the evaluators evaluated a pre-defined list of questions and answers produced by the system under approximately identical environmental conditions.

Word	Cat	a	b	c	d	e	All
stopped	V	100	66.67	100	100	87.50	90.00
moving	V	100	100	100	100	100	100
none	D	100	66.67	100	100	100	95.00
forward	D	100	80.00	100	–	100	94.12
backward	D	100	100	–	–	100	100
none	H	100	70.00	100	100	100	94.00
left	H	100	100	100	100	57.14	90.00
right	H	100	100	–	–	100	100
none	M	100	80.00	100	100	100	96.34
slowly	M	100	100	100	–	100	100
in front of	R	63.89	33.33	75.00	55.88	71.43	58.89
to the left of	R	77.78	100	100	100	100	92.31
behind	R	64.29	77.78	63.64	57.14	36.36	59.62
to the right of	R	83.33	100	66.67	100	–	85.71

Table 5.1: *J48-Simple*: accuracy per individual words

You may notice that some fields contain a $-$ rather than a numeric value. These values are missing because that word was not generated by the system during that evaluator's session. For the purpose of correlation, we decided to replace the missing values with the means of the remaining values for that word, thus the scores of other evaluators in rows rather than the scores of the same evaluator in columns. As explained below, this equals the second variable in each correlation step. Since the function to which the variables are compared is $f(x) = x$, this means that two such values will be perfectly correlated.

The degree of statistical connectedness between two variables is measured by a linear correlation coefficient. A perfect linear relation between the variables is described by a linear function. In this case the value of a linear correlation coefficient is 1 or -1. If the relationship between the variables is random, the correlation coefficient is 0. A value between the two extremes tells us how well the variables fit a linear function. The sign of the correlation coefficient defines whether the linear relationship is positive or negative, whether the values of the second variable increase or decrease by increasing

Word	Cat	a	b	c	d	e	All
moving	V	100	83.33	81.82	100	100	94.12
turning	V	84.62	66.67	89.47	–	–	82.93
stopped	V	100	85.71	100	100	100	97.92
reversing	V	100	–	–	–	–	100
none	D	100	58.33	88.89	100	100	90.91
backward	D	75.00	–	100	–	100	88.89
forward	D	100	100	94.44	100	100	98.36
spot	D	100	0	100	–	–	88.89
none	H	100	61.54	96.97	100	100	94.50
left	H	75.00	100	80.00	–	87.50	85.29
right	H	100	0	100	–	83.33	90.00
clockwise	H	100	–	100	–	–	100
none	M	100	62.50	97.30	100	100	94.53
slowly	M	100	100	100	100	100	100
gently	M	–	–	100	–	–	100
to the right of	R	92.86	33.33	91.67	66.67	42.86	77.22
in front of	R	82.35	40.00	21.05	64.71	45.45	53.61
to the left of	R	60.00	100	77.78	75.00	66.67	72.22
behind	R	33.33	54.55	18.52	69.44	23.53	42.45
facing	R	100	100	–	–	–	100
far from	R	100	–	100	–	–	100
close to	R	100	–	100	100	–	100

Table 5.2: *J48-All*: accuracy per individual words

the values of the first one.

Statistical literature describes a few linear coefficients (see Sagadin, 2003, Chapter 8 and Wonnacott and Wonnacott, 1990, Chapter 15) depending on the type of the variables that are correlated. The two most frequently used are *Pearson's product moment coefficient* (r or r_{xy}) and *Spearman's coefficient of rank correlation* (ρ or Rho) which we also use here. Pearson's product moment correlation coefficient (Sagadin, 2003, page 110ff.) determines linear correlation between two variables that are measured on a continuous numeric scale, thus interval and ratio variables. There are two requirements for this coefficient to be used. The relationship between the variables must be linear and the values of the variables must be normally distributed. The distributions of values in Table 5.1 and 5.2 appear to be unimodal, however they are not symmetric since both datasets show a high frequency of values close to 100%. One reason for this is that the sample sizes are very small: $n = 14$ and $n = 22$ respectively. This has a negative influence on the linear relationship between the variables.

Pearson's' r_xy can only evaluate a relationship between two variables. However, we would like to measure the inter-rater agreement between 5 evaluators. This can be estimated by correlating the scores of one evaluator with the average scores of all remaining evaluators, and repeating for all participants. It is thus a form of cross-correlation.[2] Each correlation coefficient expresses the strength of relationship between one evaluator and the rest of the group. The agreement within the group is estimated by taking the average of these coefficients. Perhaps a better method would be to calculate *multiple correlation* (Sagadin, 2003, page 123) where we find a correlation coefficient between a dependent variable and many independent variables in one step. A disadvantage of this approach is that the calculations for more than three or four variables can get quite complex.

Table 5.3 shows the Pearson's correlation coefficients r_{xy} obtained at each fold of

[2] I thank Rada Mihalcea for suggesting this method.

correlation for both sets of classifiers. The last column contains the average correlation coefficient. The asterisks indicate the statistical significance levels of the coefficients obtained by a two-tailed t-test (Sagadin, 2003, page 285). * indicates that the correlation is significant at the 0.05 level, and ** indicate that it is significant at the 0.01 level. "ns" indicates that the correlation is not significant.

Configuration	a:rest	b:rest	c:rest	d:rest	e:rest	Mean
J48-Simple	0.824**	0.382 ns	0.787**	0.907**	0.636*	0.707
J48-All	0.504*	0.048 ns	0.635**	0.756**	0.662**	0.521

Table 5.3: Pearson's product moment correlation coefficients

The data in Table 5.1 and 5.2 represent samples of word utterances generated by the system and the correlation coefficients r_{xy} in Table 5.3 are estimations based on these samples. The t-test is used to establish whether the sample correlation coefficients are significantly different from zero when the entire population of utterances is considered. It therefore confirms that there is evidence of association between two variables in general. The levels of significance indicate the risk that the conclusion may be wrong, that the population correlation coefficients may nonetheless be zero.

Returning to the values in Table 5.3 we can see that except for the evaluator *b* there exists a moderate to high correlation between the scores of an individual evaluator and the mean scores of the rest of the group. The average correlation coefficient for the *J48-Simple* configuration is greater (0.707) than the average correlation coefficient for the *J48-All* configuration (0.521). All correlation coefficients, except in the case of evaluator *b* are statistically significant at the level $\alpha = 0.05$ or less which means that the risk of incorrectly identifying correlation in the population is 5% or less.

What about evaluator *b*? There is a low positive correlation between the scores of this evaluator and the rest of the group in the case of the *J48-Simple* configuration and there is almost no correlation between *b* and the rest when the *J48-All* configuration is used. It is difficult to speculate about the reasons of this outcome. The occurrence

of no linear correlation between evaluator b with both configurations and strong positive correlation between every other participant and the rest of the group suggest that evaluator b should be taken as an outlier.

Note that all coefficients presented in Table 5.3 reflect the bias of b because its scores affect the mean of the group. To show the effects of exclusion of b on the correlation coefficients, we recalculated them without the scores of this evaluator. They are shown in Table 5.4. The mean scores of the coefficients increased to 0.819 and 0.725 for the *J48-Simple* and *J48-All* configurations respectively and all the coefficients are statistically significant at the level $\alpha = 0.01$. The only coefficient that decreased after excluding the evaluator b is *d:rest* in case of *J48-Simple*.

Configuration	a:rest	c:rest	d:rest	e:rest	Mean
J48-Simple	0.841**	0.862**	0.821**	0.752**	0.819
J48-All	0.608**	0.755**	0.826**	0.712**	0.725

Table 5.4: Pearson's product moment correlation coefficients excluding evaluator b

We also calculated Spearman's rank correlation coefficient or Spearman's ρ[3] (Sagadin, 2003, page 156,ff.) on the data in Table 5.1 and 5.2. Spearman's ρ is determined in the same way as the Pearson's coefficient, the only difference is that it is determined from ranked or ordinal data (R_x and R_y) rather than interval or ratio data (x and y). This makes it a non-parametric test. It is standardly used on data which cannot be measured on a continuous numeric scale and which therefore does not follow a frequency distribution. The approach assumes that its ranks do.

The data is assigned to ranks such as 1, 2, 3... r according to an arbitrary monotonic function. When arranging items to ranks it may happen that we encounter duplicate items which should be assigned the same rank. This is achieved by assigning them a tied rank which is calculated by taking the mean value of the ranks that they would

[3]The Spearman's coefficient can be written either as ρ_S or r_S if the general rule in statistics of writing population parameters with Greek letters and sample parameters with Latin letters is maintained.

otherwise jointly occupy on the ranked list. For example, if item 25 occurs twice and competes for ranks 2 and 3, we assign both items rank 2.5. The next item is assigned rank 4 and so forth. Table 5.5 shows Spearman's ρs calculated from the ranks of evaluator accuracies from Table 5.1 and 5.2.

Configuration	a:rest	b:rest	c:rest	d:rest	e:rest	Mean
J48-Simple	0.513 ns	0.173 ns	0.709**	0.610*	0.520 ns	0.505
J48-All	0.355 ns	0.080 ns	0.212 ns	0.679**	0.589**	0.383

Table 5.5: Spearman's rank correlation coefficients

When comparing these coefficients with Pearson's coefficients in Table 5.3 it can be seen that all of their values, with the exception of (J48-All, b:rest), are consistently lower. This is graphically shown in Figure 5.1 and 5.2. The average Spearman's correlation coefficient is 0.505 for the *J48-Simple* configuration and 0.383 for the *J48-All* configuration. These values indicate moderate to low positive correlation. In comparison, the average Pearson's correlation coefficients based on the same same data are 0.707 and 0.521 respectively. Fewer coefficients are statistically significant as confirmed by a two-tailed t-test.

The same pattern emerges if we exclude the ratings of evaluator b as shown in Table 5.6. All coefficient values are consistently lower from the Pearson's coefficients in Table 5.4. When compared to the same coefficients in Table 5.5 their values are higher and statistically more significant, except for (J48-All, e:rest). This confirms once again that the judgements of evaluator b are different to the rest of the group.

Configuration	a:rest	c:rest	d:rest	e:rest	Mean
Simple	0.667**	0.789**	0.667**	0.622*	0.686
All	0.446*	0.430*	0.755**	0.499*	0.533

Table 5.6: Spearman's rank correlation coefficients excluding evaluator b

Sagadin (2003, pages 162–163) states that in general ρ_S is lower than r_{xy} when calculated on the same data. This is because Spearman's coefficient is calculated only from

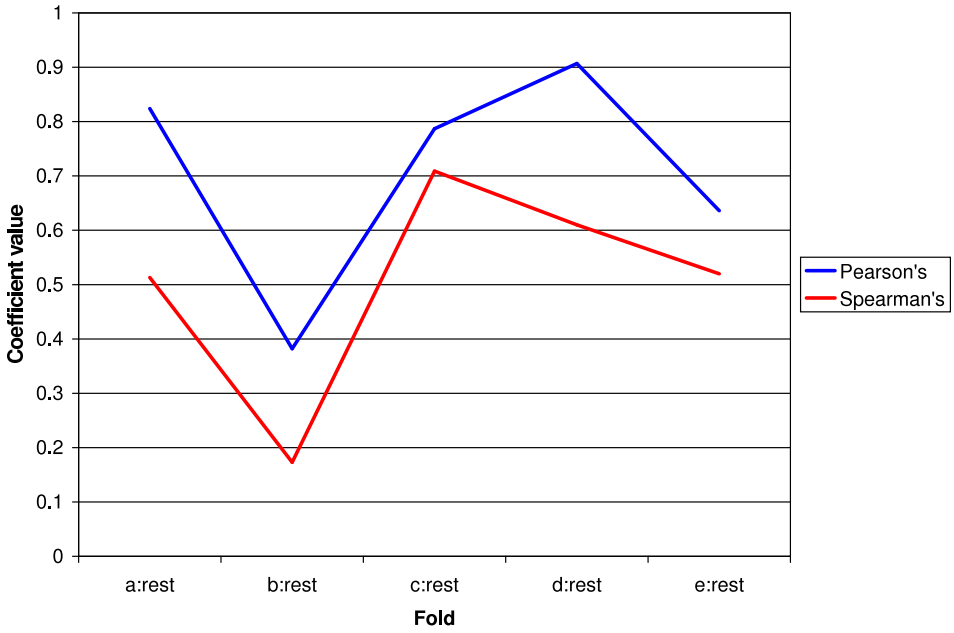


Figure 5.1: *J48-Simple:* evaluator agreement as correlation coefficients per fold

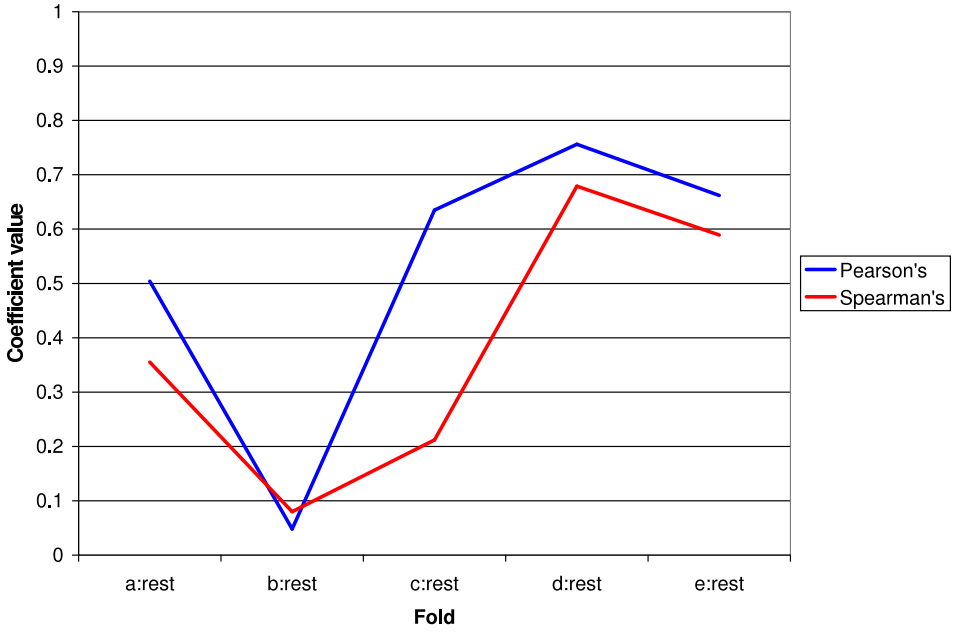


Figure 5.2: *J48-All:* evaluator agreement as correlation coefficients per fold

the ranks rather than the actual interval or ratio values. The advantage of ρ against r_{xy} is that it reduces the effect of extreme values when these are converted to ranks. Pearson's coefficient thus gives a better estimation of correlation for data that can be measured on a continuous numeric scale. However, such data is not always available in which case ρ_S must be used exclusively.

While discussing the κ coefficient, Artstein and Poesio (2005) point out referring to examples and discussion in Bartko and Carpenter (1976) that correlation coefficients are not sufficient measures of agreement. They measure association rather than agreement and the two measures are not quite the same. To illustrate this point they give an example of a situation where one evaluator consistently marks all items higher than the other and the relation between the scores of both evaluators corresponds to a linear function. For example, they evaluate the first item (1, 1), the second (2, 4), the third (3, 6) and so on. Such evaluators disagree on all items, but the correlation coefficients would predict a perfect correlation of 1.

This does not represent a difficulty in our case because we are not using correlation to measure direct agreement of two evaluators on a set of items. As previously discussed this is not possible because of the nature of our data. We chose to measure agreement as association of accuracy scores per word between an evaluator and the rest of the group. Under this scenario the effect is working to our advantage. If an evaluator consistently agrees to fewer or more generations of all words than the rest of the group then such an evaluator is biased. Such bias correctly has no effect on correlating the accuracy scores.

To conclude, the statistical tests show that there is an association between the accuracy scores of individual words between each evaluator (excluding evaluator b) and the rest of the group. We can confirm the two questions with which we started the discussion in this section: (i) the human evaluators do work as a single body to which the performance of the system can be compared, and (ii) the system has not been tuned

to particular humans but captures some universal knowledge.

5.2.3 Classifier performance and system performance

Let us now turn to examining the evaluators' ratings of system performance. Table 5.1 and 5.2 give their accuracy ratings per individual words. Table 5.7 and 5.8 give the same ratings per individual category. The last column of both tables labelled J48 includes the estimated accuracies of the classifiers that the system was using in the evaluation taken from Table 3.7, page 105. Figure 5.3 and 5.4 show the data from Table 5.7 and 5.8 graphically.

How do the results from both evaluations compare? The classifier accuracies are the average accuracies obtained through a 10-fold cross-validation (Section 3.5.1). The reported accuracy is the ratio between the number of correct classifications over the total number of classifications. In system evaluation the accuracy is determined on an independent test set so to speak. It is the ratio between the number of times a human evaluator agreed with the generated description over the total number of generated descriptions. There is a slight difference between the two situations in how a positive match is made. In classification the correct value of the class is pre-defined and hidden from the classifier and this is matched with the predicted class. In system evaluation an evaluator hears the generated description before they give their evaluation. In this respect it is possible that the system biases the evaluator.[4] Most importantly, both evaluations do not evaluate quite the same thing. Classifier accuracy evaluates only the target class predicted by the classifier. On the other hand, system accuracy evaluates classifier accuracy relative to other choices the system makes when generating a description (Section 4.4.2 and 4.4.4). For example, in the case of descriptions of object relations this includes the method by which the objects are chosen.

[4]The bias may be encouraged by the input method. If the evaluator agrees with the generated description, they answer "yes". If they do not, they answer "no" and provide an alternative description.

Category	a	b	c	d	e	All	J48
Motion	$n = 36$	$n = 17$	$n = 14$	$n = 2$	$n = 21$	$n = 90$	
Verb	100	88.24	100	100	95.24	96.67	89.02
Direction	100	76.47	100	100	100	95.56	87.80
Heading	100	82.35	100	100	85.71	93.33	97.56
Manner	100	82.35	100	100	100	96.67	70.73
Relation	$n = 65$	$n = 23$	$n = 19$	$n = 53$	$n = 22$	$n = 182$	
Relation	67.69	65.22	68.42	66.04	59.09	65.93	75.90

Table 5.7: *J48-Simple:* system performance and classifier performance

Let us first turn to the evaluated accuracies of the system using the *J48-Simple* set of classifiers (Table 5.7 and Figure 5.3). The scores of the system performance from evaluator *b* on the descriptions of motion (the categories *Verb*, *Direction*, *Heading* and *Manner*) are lower than the scores of other evaluators. As discussed in the previous section this is not due to the fact that evaluator *b* is not correlating or "agreeing" with other evaluators. Evaluator *b* could have had consistently lower scores from other evaluators and still would have been in perfect agreement with them. The grey line in the diagram marks the accuracies of the classifiers. Evaluator *b* judged the performance of the system on the motion categories *Verb*, *Direction* and *Heading* lower than the accuracy of the classifiers producing them. On the other hand evaluators *a*, *c* and *d* agreed with all motion words generated by the system and hence their lines overlap at 100 per cent. They consider the performance of the system on the descriptions of motion better than the performance of the classifiers which the system was using. A similar trend is observed with evaluator *e* on the categories *Verb*, *Direction* and *Manner* but not on the category *Heading*. The black line which represents *all* evaluators confirms that on the motion categories evaluators tend to judge the performance of the system better than the accuracy of the classifiers underlining it. This is particularly evident on the *Manner* category where the classifier performs worst but evaluators judge this category similar to other categories – the black line is almost flat. Overall, the accuracies

for descriptions of motion are very high both for the system and for the classifiers. For example, the mean accuracy over all categories for the body of evaluators (the column *all* in Table 5.7) is 95.56% whereas the classifier accuracy is 86.28%.

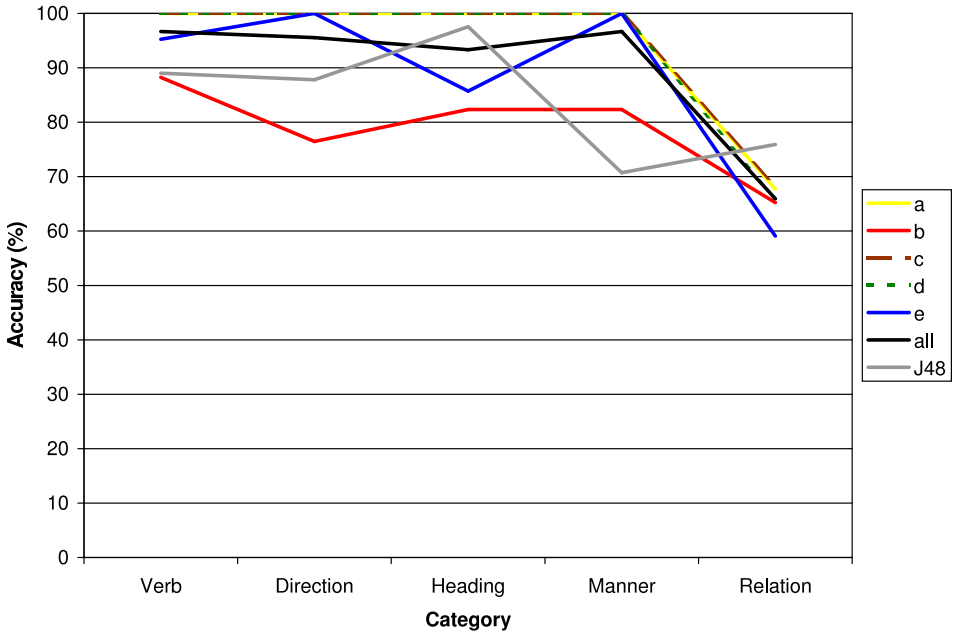


Figure 5.3: *J48-Simple:* system performance and classifier performance

The trend observed with the categories describing motion is not replicated with the relation category. The classifier can predict the correct relation in 75.90% of cases but the evaluators considered the system to generate a good description in 65.93% of cases, thus a decrease of almost 10%. There is not much variation in the judgements of different evaluators as their values cluster at that value.

To summarise, human evaluators judged that the system performs better than its underlying classifiers for descriptions of motion and worse for descriptions of object relations. In the latter case, the system generates a correct description in 2/3 of cases. This is not an ideal but still acceptable level of performance given the nature of the task.

Let us now turn to the performance of the system that uses the *J48-All* configuration of classifiers (Table 5.8 and Figure 5.4). The graph shows many of the same trends as the

Category	a	b	c	d	e	All	J48
Motion	$n = 53$	$n = 22$	$n = 53$	$n = 7$	$n = 41$	$n = 176$	
Verb	96.23	77.27	88.68	100	100	92.61	48.22
Direction	96.23	72.73	92.45	100	100	93.18	55.68
Heading	98.11	68.18	92.45	100	95.12	92.05	60.77
Manner	100	72.73	98.11	100	100	96.02	54.70
Relation	$n = 66$	$n = 28$	$n = 72$	$n = 110$	$n = 58$	$n = 334$	
Relation	72.73	57.14	44.44	70.00	43.10	59.28	69.12

Table 5.8: *J48-All:* system performance and classifier performance

one for the *J48-Simple* configuration in Figure 5.3. Evaluators *a*, *c*, *d* and *e* consider the performance of the system on motion words between 90 and 100%. Evaluator *b* again stands out with much lower scores than the rest of the group but these are now higher than the accuracy of the underlying classifier. The classifier does again least well on the *Manner* category (56.30%) , but this is not reflected in the evaluator's judgements of the system's performance (96.02%).

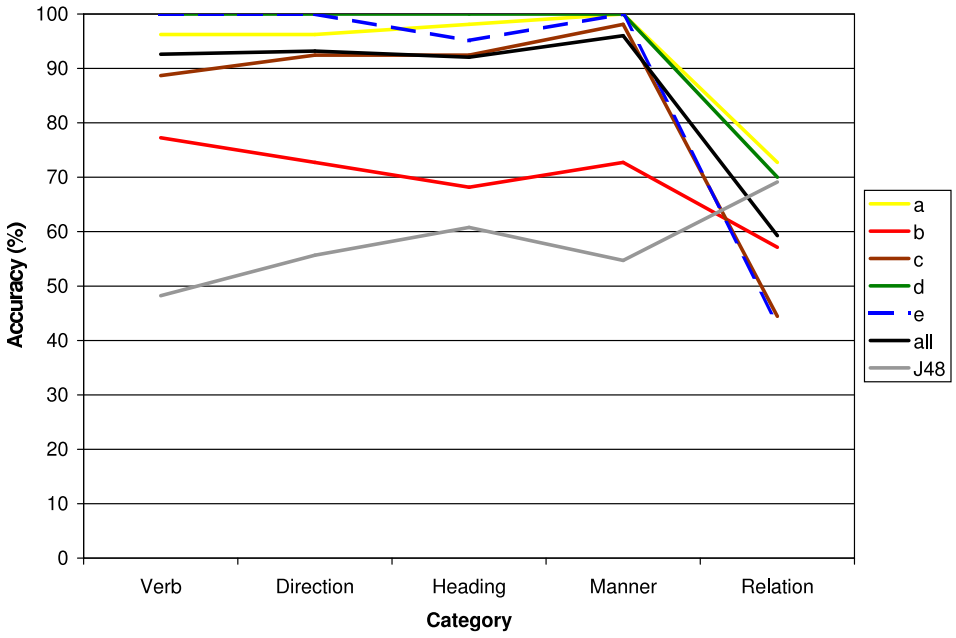


Figure 5.4: *J48-All:* system performance and classifier performance

There is less consistent behaviour in the judgements of performance on descriptions

of object relations compared both to the motion descriptions of the same classifier configuration and descriptions of object relations of the *J48-Simple* configuration described previously. Our baseline, the accuracy of the classifier, is 69.12%. Evaluators *a* and *d* judge the performance to be slightly better than the baseline, while evaluators *b*, *c* and *e* judge it to be worse. Evaluators *c* and *e* particularly stand out as their evaluation figures are as low as 44.44% and 43.10% respectively. If all evaluators are considered together (*all*), the evaluated performance of the system is 59.28%, thus again a good 10% lower than the baseline.

The performance of the system using the *J48-All* configuration of classifiers is thus similar to the performance of the system using the *J48-Simple* configuration. Human evaluators perceive that the system is performing better than its underlying classifiers on descriptions of motion, but they consider it to perform slightly worse than the classifier on descriptions of object relations.

A plausible explanation of this outcome is that categories of motion may contain words that are semantically less restrictive than words locating objects: they are synonyms. For example, the category *Verb* contains words such as "going", "moving" and "continuing" which have a very similar reference for a human but not for a machine learner. Therefore, there is a higher chance that the generated description will be acceptable. The categories *Direction*, *Heading* and *Manner* contain words with clearer semantic devisions but all of them contain the word "none", a dummy word that was assigned to a category if no other word of that category was found in the pre-learning description. We argued that for the purposes of machine learning a null word captures better the semantics of a description than if the category is marked to contain a missing value (Section 3.3.3, page 71). Not only does "none" have a distinct meaning, it has a default meaning and an anaphoric meaning. For example, for the *Direction* category "none" has the same meaning as "straight". On the other hand "none" can also be interpreted to refer anaphorically to the previous description of *Direction* in the

discourse.

Another explanation of the evaluation results is that learning and generating descriptions of object relations is more complex than learning and generating descriptions of motion. This means that our learning (Section 2.4.2.2) and generation models (Section 4.4.4) for descriptions of object relations capture human knowledge less well than the models for descriptions of motion. We return to the qualitative evaluation of the system's performance and its shortcomings in Section 5.2.5. As we pointed out at the beginning of this section, generating descriptions in general requires steps in addition to classification. For this reason a drop in the system's performance is quite expected.

Overall, the figures in Table 5.7 and 5.8 indicate the humans evaluated the system quite favourably which means that it has reached a good level of performance. Therefore, we can confirm that machine learning is able to capture the semantics of spatial descriptions.

5.2.4 System performance excluding chance

We argued that accuracy is not a good indicator of performance because the agreement between the two standards of comparison, in this case the system and the evaluator, could be due to chance. Chance is measured as a bias of each standard toward assigning each category, in this case generating a particular word (Section 3.5.3).

Table 5.9 and the associated Figure 5.5 and 5.6 represent the performance of the system according to the estimated values of the κ coefficient per each evaluator-system pair. They also include the estimated κ coefficient for the underlying J48 classifier (Table 3.8, page 110). When comparing both figures to Figure 5.3 and 5.4 from the previous section which show the performance of the system according to accuracy we see that both sets of figures reveal identical trends. The differences in κ values both across different categories judged by one evaluator (the horizontal dimension) and across different evaluators on the same category (the vertical dimension) appear to be

Category	a:s	b:s	c:s	d:s	e:s	all:s	J48
			J48-Simple				
Verb	1	0.8007	1	1	0.9417	0.9523	0.7738
Direction	1	0.6543	1	1	1	0.9312	0.7758
Heading	1	0.7241	1	1	0.7838	0.8939	0.9479
Manner	1	0.6988	1	1	1	0.9260	0.2523
Relation	0.6119	0.5593	0.5935	0.5858	0.4649	0.5730	0.6747
			J48-All				
Verb	0.9397	0.6504	0.8264	1	1	0.8959	0.2852
Direction	0.9407	0.6086	0.8855	1	1	0.8934	0.2960
Heading	0.9670	0.5048	0.8696	1	0.9251	0.8706	0.3432
Manner	1	0.5661	0.9530	1	1	0.9229	0.0859
Relation	0.6786	0.4599	0.3035	0.6306	0.2747	0.4965	0.6110

Table 5.9: The κ values per category for each evaluator-system pair

more pronounced than the differences in accuracy. The same also holds if we compare the performance of the J48 classifier with the evaluated system performance.

Following the discussion in Section 3.5.3 the interpretation of the κ coefficient depends on the nature of the task. In classification tasks any value of κ greater than 0 is a positive result since it indicates that a classifier is performing better than a baseline classifier which is assigning classes randomly or is only assigning the majority class (page 111). The same also holds for a system generating descriptions. The mean κ value for the group of evaluators [a...e] and the system in the *J48-Simple* scenario is 0.8553, $s = 0.1424$ and the mean κ value for the same pair in the *J48-All* scenario is 0.8159, $s = 0.1605$. According to the literature the evaluator and the system show a good measure of agreement ($\kappa > 0.8$) which is also considerably greater than the agreement reached by the underlying classifier (*J48-Simple:* $\bar{\kappa} = 0.6849$, $s = 0.2335$, *J48-All:* $\bar{\kappa} = 0.3243$, $s = 0.1684$).

The distribution of values is not even across all categories. The agreement $\kappa > 0.8$ is only present on the motion categories *Verb*, *Direction*, *Heading* and *Manner* in both *J48-Simple* and *J48-All* scenarios, whereas the agreement on the *Relation* category is

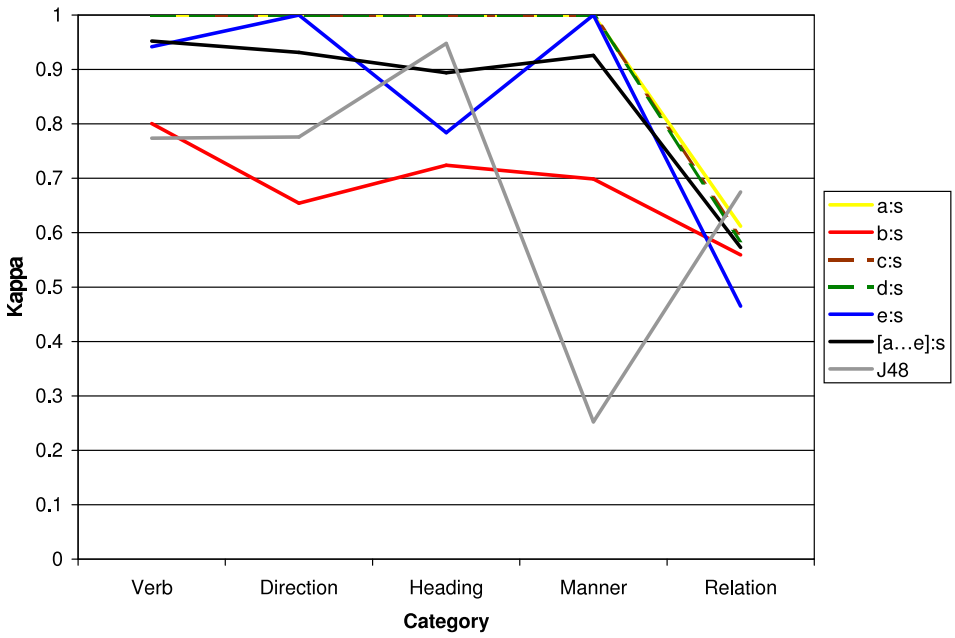


Figure 5.5: *J48-Simple:* κ per category for each evaluator-system pair

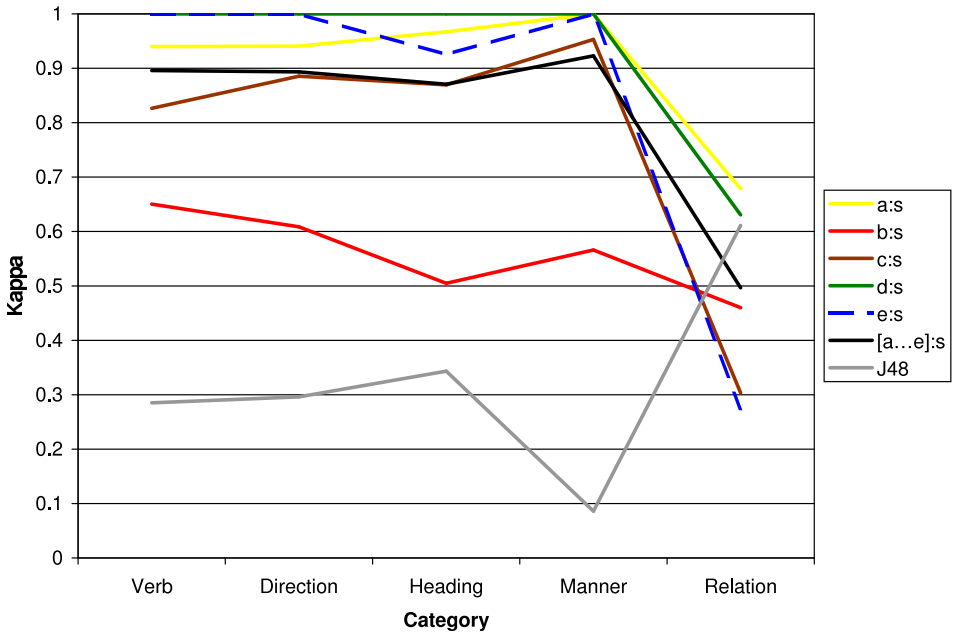


Figure 5.6: *J48-All:* κ per category for each evaluator-system pair

markedly lower: 0.5730 and 0.4965 respectively. In the same way as accuracy (page 195) these κ values are lower by approximately 0.1 from the κ values of the underlying J48 classifier. The κ coefficients therefore confirm the conclusions in the preceding section. They also favourably indicate that the system is performing well beyond chance.

To calculate the κ coefficient in Table 5.9 we had to estimate probabilities with which human evaluators and the system generate a word of a particular category (Equation 28, page 109). The product of these probabilities summed over all words in the category represents the estimate of the agreement by chance (A_e) (Equation 27, page 109)[5] given in Table 5.10. The estimated probability with which the system assigns a word to a category is constant in the calculations of A_e across different evaluators. It follows that the values of A_e when compared between different evaluator-system pairs reveal the bias of each individual evaluator. We plot them in Figure 5.7 and 5.8. The figures also include the plots of observed agreement (A_o) or accuracy (from Table 5.7 and 5.8) and κ (from Table 5.9).

Category	a:s	b:s	c:s	d:s	e:s	all:s
			J48-Simple			
Verb	0.4239	0.4099	0.3837	0.1156	0.1834	0.3022
Direction	0.3581	0.3194	0.3322	0.3300	0.4166	0.3546
Heading	0.3963	0.3603	0.3964	0.3719	0.3391	0.3711
Manner	0.6325	0.4141	0.7043	0.5140	0.5319	0.5503
Relation	0.1675	0.2108	0.2231	0.1802	0.2355	0.2021
			J48-All			
Verb	0.3744	0.3498	0.3479	0.1412	0.2370	0.2900
Direction	0.3639	0.3033	0.3408	0.3281	0.4475	0.3604
Heading	0.4278	0.3575	0.4208	0.3831	0.3484	0.3857
Manner	0.5515	0.3715	0.5979	0.4480	0.4808	0.4838
Relation	0.1515	0.2064	0.2023	0.1879	0.2155	0.1913

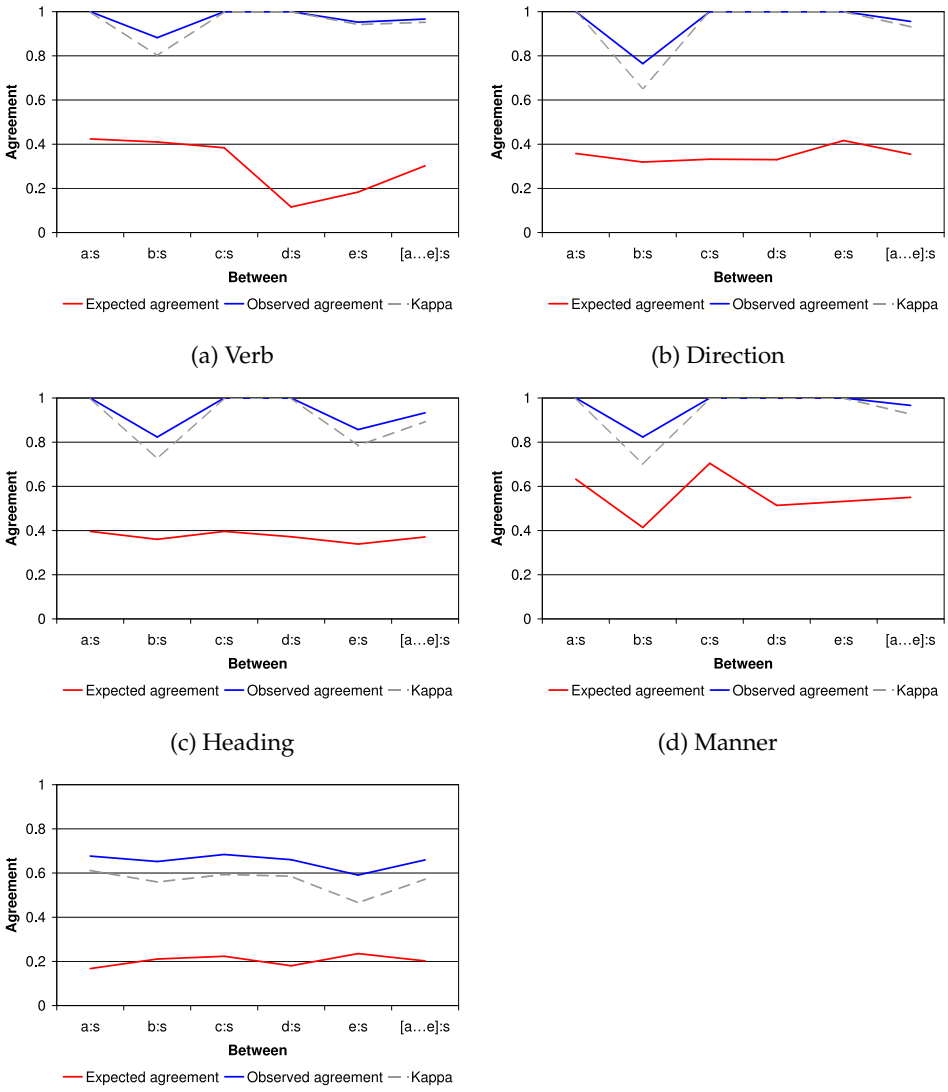
Table 5.10: The expected agreement per category for each evaluator-system pair

If we compare A_e between different evaluators and the system for both *J48-Simple*

[5]The equation assumes that the system and evaluators assign a word independently of each other. We mentioned on page 193 that this condition may not be satisfied completely. In such cases A_e are underestimated.

and *J48-All* configurations of the system, we observe that in many cases there is not much difference between one evaluator and another: the lines are more or less flat. This means that these evaluators distribute words into categories in a similar way which is a positive result. There are a couple of differences though. For example, on the *Verb* category in both *J48-Simple* and *J48-All* scenarios A_e is markedly lower than the A_e for other evaluators-system pairs between (i) the system and evaluator *d* (0.1156 and 0.1412), and (ii) the system and evaluator *e* (0.1834 and 0.2370). A similar drop is observable on the *Manner* category for the *b:s* pair (0.4141 and 0.3751). On the other hand there is also an increase in agreement by chance between the system and evaluator *c* on the same category (0.7043 and 0.5979). A drop in A_e for a particular evaluator compared to other evaluators means that the evaluator is biased toward different vocabulary than other evaluators, an increased agreement by chance indicates that the evaluator tends to follow the system more than other evaluators. Thus, observed at the level of individual evaluators, the agreement by chance tells us about the idiosyncrasies of particular subjects when evaluating descriptions. We can expect similar idiosyncrasies also occur when a corpus of descriptions is collected for machine learning.

Note that almost identical patterns of A_e are observable in both *J48-Simple* and *J48-All* scenarios. This is expected, since changing the scenario, the underlying classifier that the system is using, does not affect evaluators in any way nor were they explicitly told that there was a difference in the system. The actual values of A_e between the two scenarios are also very similar. In the experiment it could not be ensured that both coders used the same tag set and this is one of the ways in which our usage of κ diverges from a standard tagging task. We can be sure that the classifiers from the *J48-All* scenario capture most of the words used to generate spatial descriptions since their corpus was created with no restrictions in vocabulary and from five describers. In the *J48-Simple* scenario, the system only has a knowledge of the basic spatial words, whereas the evaluators judged its performance using their complete linguistic com-



(a) Verb

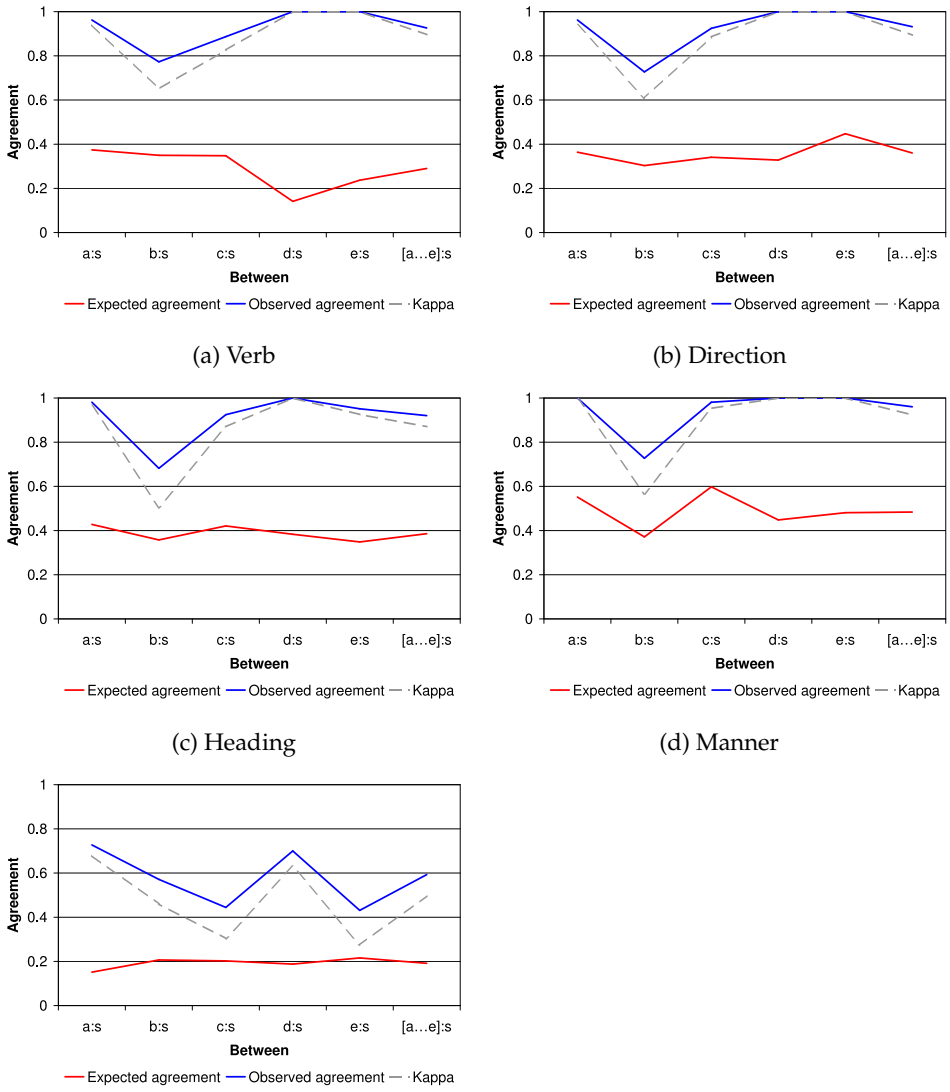
(b) Direction

(c) Heading

(d) Manner

(e) Relation

Figure 5.7: *J48-Simple:* expected agreement, observed agreement and κ per category for each evaluator-system pair



(a) Verb

(b) Direction

(c) Heading

(d) Manner

(e) Relation

Figure 5.8: *J48-All:* expected agreement, observed agreement and κ per category for each evaluator-system pair

petence. Thus, even if evaluators and the system distribute words uniformly within each category, in the *J48-Simple* scenario the probability mass of an evaluator choosing a word is spread out across more words compared to a situation where the system is choosing a word simply because evaluators operate with more words. There is an *a priori* bias. Consequently, the agreement by chance between the two is expected to be lower than in the *J48-All* scenario. Examining Table 5.10, the differences between the A_e coefficients are minimal. Performing a simple count we can conclude that there are 60 coefficients in the table. Of these, 12 are lower in the *J48-Simple* than in the *J48-All* scenario and 18 are higher. This means, that the basic vocabulary selected for the *J48-Simple* scenario is also the most frequently used vocabulary in the *J48-All* scenario to which most probability mass is assigned. We made an identical conclusion in Section 3.5.6.

5.2.5 Qualitative evaluation of the system's performance

In this section we turn to the qualitative evidence for the performance of the *pDescriber* system. This evidence consists of comments and observations of errors made by the evaluators when interfacing with a working system.

5.2.5.1 Ambiguity of heading and direction

A few evaluators pointed out that the descriptions of direction such as "left" and "right" are ambiguous when used to describe motion. "Moving right" can mean moving forward with a heading in the clockwise direction. It can also mean making a sharp turn to the region that is to the right of the current location. This usage is thus identical to the usage of "right" when describing object relations. Both types of movement are exemplified in Figure 5.9.

One would expect that the classifiers in machine learning would internalise the distinction between the two usages provided that the training data is consistent and there is enough of it. However, this was not the case. Perhaps the projective usage is

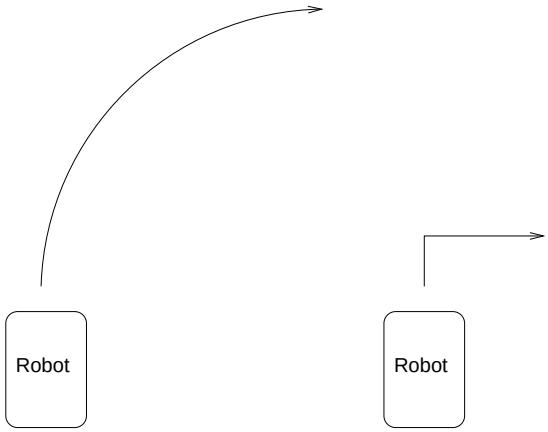


Figure 5.9: Two meanings of "right"

too complex to be learned from the available odometry parameters. The interpretation is also driven by the accompanying verb. For example, if the verb is "edging", the adverbial interpretation of direction is more likely. On the other hand, if the verb is "turning", the projective interpretation is preferred. Because the system treats each linguistic category as a separate target concept, such tendencies are not preserved in the knowledge learned. The consequences of this are that the semantic distinction is lost, and that the system may over-generate descriptions since certain adverbs are used with verbs that are less likely to occur with in human descriptions.

A similar case is a description "moving backward" which either means (i) that the vehicle is reversing or moving in the direction that is behind its back (Figure 5.10a); or (ii) that it has reversed, but is now moving forward to the direction that was previously behind its back (Figure 5.10b). The latter interpretation is more complicated since it involves an action of rotating the vehicle by 180°. To learn such a description, the learner would have to abstract over a chart of actions rather than over physical descriptions of the environment. Therefore the description cannot be learned by the present system. Since the scene in Figure 5.10b can be described either with "moving forward" or "moving backward" and the system cannot take into account complex actions, it

may rarely over-generate "moving backward" in cases where only "moving forward" is applicable.

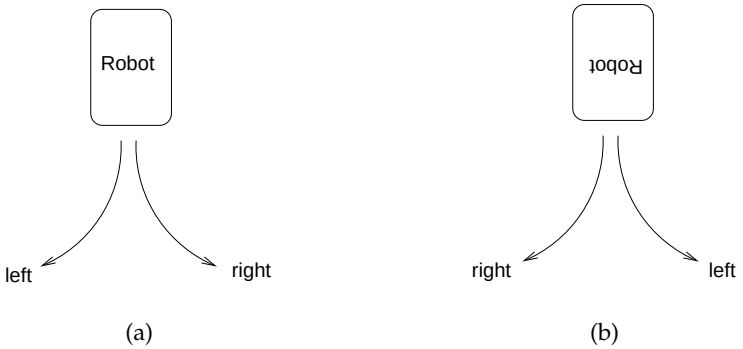


Figure 5.10: "Moving backward" or "moving forward"

Both Figure 5.10a and Figure 5.10b indicate an intrinsic reference frame for the descriptions "left" and "right". It was noticed that for the scene in Figure 5.10a descriptions of heading were also generated by the robot in respect to the relative reference frame fixed on an observer facing the front of the robot in which case "left" and "right" are reversed. When a scene in Figure 5.10b was described with "moving forward" the intrinsic reference frame was chosen in most cases.

5.2.5.2 Object shape

Our models for learning descriptions of object relations do not take into account the shape of the objects. On the SLAM map each object is represented as a point defined by its centre. While this works reasonably well for most of the objects, difficulties arise with objects which are not square-shaped and are markedly different in one dimension such as "the wall" and "the barrier". Figure 5.11a shows a setting which the robot non-intuitively describes with "The wall is in front of Flakey" and "The chest is behind the wall". When one considers that the system only perceives the wall as a point represented by the black circle, it becomes clear why such descriptions are generated.

There were only a couple of such objects in our environment, most notably the walls, for which the system generated erroneous descriptions. It may appear that rep-

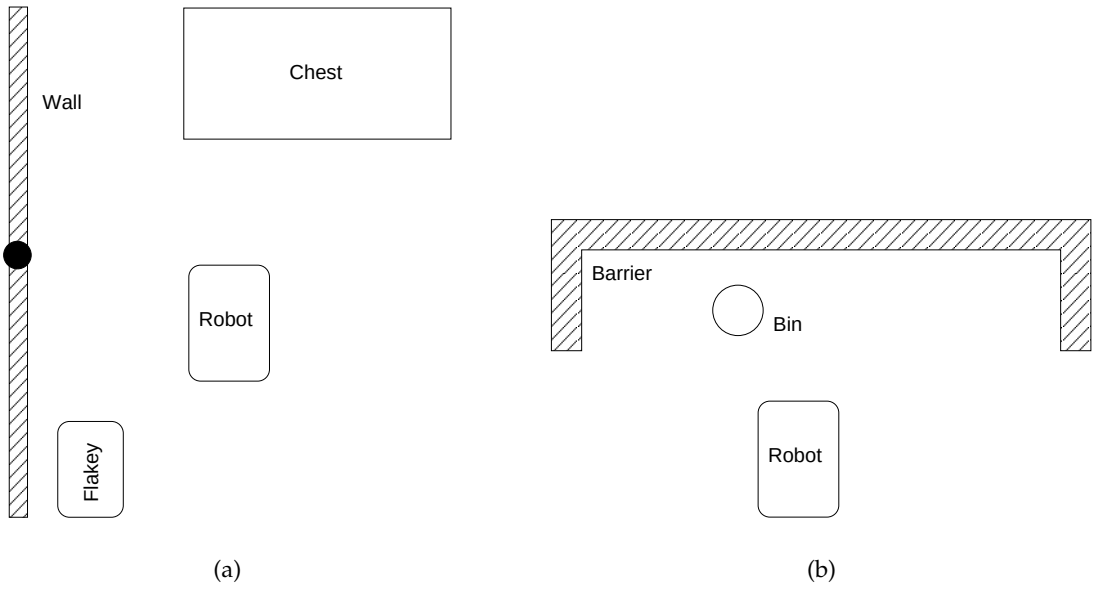


Figure 5.11: Where is the wall, the chest, the bin?

resenting objects as a convex hull of their points in a two-dimensional space would solve the problem. However, the effects of object shape on descriptions are even more complex than captured by this model as exemplified by the scene in Figure 5.11b which would be described with a *complex* description of spatial relation "The bin is close to the barrier between its two ends". A system generating such descriptions should be able to generate complex descriptions and be able to refer to the parts of the objects and their properties. The issue here is not only grounding the object in the physical environment, but reasoning about the grounded objects. This is a process which takes places at a different level of meaning representation.

No doubt, to improve the performance of *pDescriber* a better treatment of object shape should be devised and integrated to it. Currently, we avoided the problem altogether as the information about the object shape is not immediately available from MOOS which only represents the environment as clouds of points. Abstraction of objects from points is a difficult problem and would require a research project of its own.

5.2.5.3 Switching the perspective

Although evaluators were told that the descriptions are generated from the perspective of the robot (Section 2.4.2.2, page 45), it was very easy for them to switch from the relative reference frame orientated relative to the robot to the intrinsic reference frame orientated relative to the reference object.

Two reasons for such behaviour can be identified. Firstly, it became apparent that such switches of perspective were already unintentionally present in the classifier training data since the system would occasionally produce descriptions according to the intrinsic reference frame. In this case the majority of evaluators would judge the description as acceptable. Only a few would point out that the description is not good if the speaker's perspective should be considered. Secondly, the properties of some objects invite human evaluators or describers to use the intrinsic rather than the relative reference frame. This is particularly the case if these objects are larger than describer (walls, barriers and cupboards), have an identifiable front and are animate (another robot).

Figure 5.12 shows two scenes, one with two robots Marge and Homer who are both facing north, and the second with the robot Marge and a table. In the first scene, the switch of the reference frame from being relative to Marge to being relative to Homer is possible and the description "the tyres are in front of Homer" is valid. On the other hand, the description "the tyres are in front of the table" to describe the second scene is unusual, because the table does not have front and back sides and is therefore not a good object to fix the orientation of the reference frame.

The scene in Figure 5.13a contains three objects: the robot, the table and the wall. If the robot is the describer and it fixes the orientation of the relative reference frame then two intuitive descriptions would be "the table is in front of me" and "the table is in front of the wall". If the intrinsic reference frame is fixed by the reference object "the wall" then both of these descriptions are equally acceptable. However, the descriptions

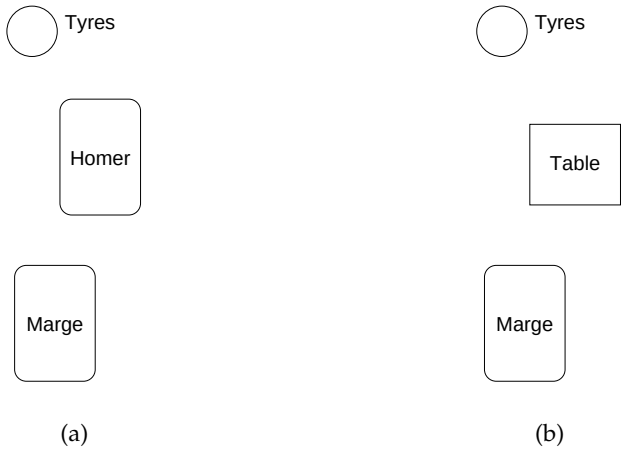


Figure 5.12: Reference frame and properties of objects

"the wall is behind the table" and "I'm behind the table" are only acceptable under one, either relative or intrinsic reference frame.

Figure 5.13b shows the same scene but with another object "a stack of tyres" placed in the proximity of the wall. The same ambiguity applies here but less successfully. A description "the table and the tyres are in front of me" is acceptable under the interpretation according to either reference frame. The ambiguity can be resolved through a blocking effect of one object better satisfying the criteria of being "in front of". If we are subsequently told that "the tyres are not quite in front of me" then we can be sure that the orientation of the reference frame is fixed by the robot. Alternatively, if we are told that "the table is not quite in front of the wall" then the orientation of the reference frame is fixed by the wall. The disambiguation is achieved by providing additional information, in the same way as subsequently saying "the tyres are behind the table".

There is one situation where a description can only be made using the intrinsic reference frame. This is when the robot is trying to describe its own location and is therefore the located object itself. This means that it cannot fix the orientation of the reference frame and consequently there must be another object to do so. The description "I'm in front of the chair" unambiguously means that the robot is located in the

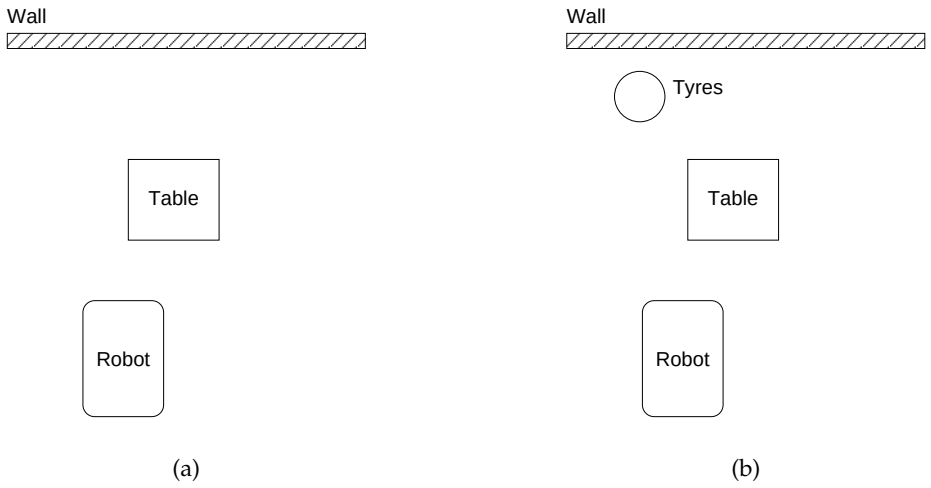


Figure 5.13: Which reference frame?

region around the seating area of the chair.

The preceding discussion shows that there is a substantial ambiguity regarding the choice of reference frame in scenes that contain three or more objects where the located object is between the speaker and another reference object. This is especially so if the latter satisfies the criteria for fixing the orientation of the reference frame better than the speaker. This ambiguity easily misleads the describer or the evaluator to switch from the intended relative reference frame to the intrinsic frame and use it also in their subsequent descriptions or evaluations.

5.2.5.4 Descriptions of objects outside the field of vision

Some evaluators did not like descriptions of objects that were not in the "vision field" of the robot. Others considered them natural. Figure 5.14 shows two such scenes. Technically, "the vision field" of the robot is much greater than that of a human observer. It includes a complete representation of the room and therefore corresponds to a mental map. Humans too use mental maps for spatial navigation and in such cases are able to refer to objects that are not in their field of vision. Perhaps then, referring to an object that cannot be seen is not completely unnatural. A description such as "the wall is in

front of the table" referring to the scene in Figure 5.14a cannot be because the system is over-generating. Typical over-generations would include sentences such as "the wall is behind the table". It must be that some describers described the objects in this way when creating machine learning datasets and such knowledge has been incorporated into the models that were learned.

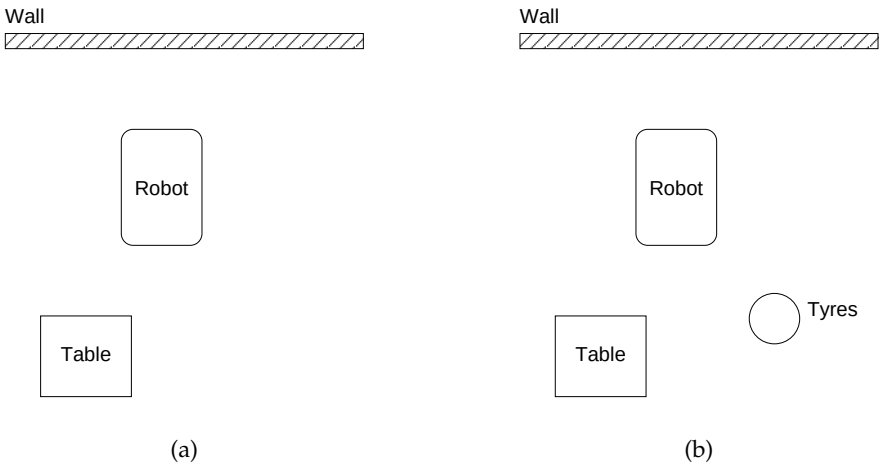


Figure 5.14: Descriptions of objects outside the field of vision

The description "the tyres are to the right of the table" referring to the scene in Figure 5.14b could be either a result of an over-generation by a human describer or an over-generation by the system.

5.2.5.5 Competing descriptions

Occasionally, *pDescriber* fails to generate the best description. When generating descriptions the system must first select a pair of objects. The description is about these objects and they are chosen because they are relevant in the current discourse. Because our system does not contain a discourse model it uses a very simple method of object selection: it chooses a pair at random. The choice of objects involves higher-level contextual reasoning and could be added to the system in the future. On the other hand, the system always attempts to choose the best description of relation between a pair of objects because this knowledge is contained in the classifiers that were learned.

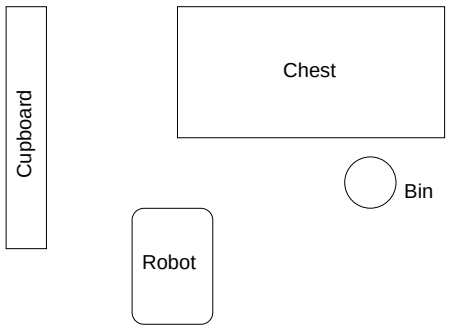


Figure 5.15: Competing descriptions

In some cases the choice of objects was indeed infelicitous. To describe the scene in Figure 5.15 the system would randomly choose "the cupboard" as the located object and "the bin" as the reference object. The classifier would predict the best description relating these objects as "left" and the description "the cupboard is to the left of the bin" would be generated. Human evaluators point out that such descriptions are acceptable but to describe the location of the cupboard, "the chest" would be a far better reference object. Descriptions in which a more salient reference object could be chosen and where the chosen reference object was outside the robot's vision field were judged to be considerably worse. This is because it is highly unusual to relate a located object to a non-salient object outside the vision field when there exists a salient object in the vision field of the speaker.

5.2.5.6 Final remarks

In the preceding discussion we summarised the shortcomings that users reported while interfacing with *pDescriber* and we attempted to provide explanations for them. In many cases this included proposing extensions and adjustments of the current system that are required for a complete system that generates spatial expressions. However, these go beyond the scope of the current project. With the exception of the perspective, the evaluators were not instructed that they should ignore evaluating a particular part

of the system's behaviour. The purpose of the experiment was to show how well the system performs in practice given the restricted representations of the state of the robot and its environment that we have access to. When the preceding observations are considered with the results in Section 5.2.3 we can conclude that the system performs remarkably well without implementing many additional features.

5.2.6 Summary

In this section we discussed the evaluation of *pDescriber* by humans. The system uses the classifiers built by machine learners to generate new descriptions of motion and object relations in real scenes. We started by outlining our experimental design. We proceeded by measuring the agreement between the evaluators. This was considered important because it shows us how reliable the judgements are and also confirms that the system has captured generalisations that are replicated between the individual evaluators. We concluded that a sufficient level of evaluator agreement was reached. In the next step we compared the performance of the system with the performance of the underlying classifiers. It turned out that according to the combined judgements of all evaluators the system was performing better than the underlying classifiers on the motion categories but slightly worse than its classifiers on the object relation category. We attributed this to the fact that object relations are more complex than motion categories and other, not just topological features may be important in their generation. We also used the κ coefficient to measure the agreement between the evaluators and the system which gives us better estimates of the true agreement that is not due to chance. Following the quantitative evaluation of the system, we concluded with a qualitative evaluation where we summarised the most typical observations that evaluators reported during the experiments. These comments point to further improvements of the system but also, when compared to the quantitative evidence, assure us that the performance of the system is indeed encouraging.

5.3 Evaluation of *pDialogue*

5.3.1 Evaluation in general

pDialogue responds to linguistic requests for motion issued by humans and answers questions about the location of objects as outlined in Section 4.5. As pointed out there, the flow of linguistic and non-linguistic information is in in many cases reversed compared to its flow when it was collected: the classifiers start with a linguistic description and predict the properties of the robot and the environment corresponding to it. These properties are not sufficient to generate a response. For example, in the case of motion they must be converted to commands that bring the robot to the state which they describe. In the case of descriptions of object relations the properties represent a region of space from which objects must be selected. One of the questions that the evaluation of *pDialogue* attempts to answer is whether the knowledge that was obtained from descriptions of states and scenes can also be used to generate motion and answer questions about objects.

In Section 4.5.1 we listed questions and requests that the system can handle as Examples 1–6, page 156. We briefly repeat them below and outline their significance in evaluation.

Questions such as "What can you do?" in (1) are answered by the component that matches words to patterns associated with pre-defined responses (Section 4.5.4). The pattern matcher was introduced to the system for practical reasons: to demonstrate some simple linguistic interaction with the robot and to deal efficiently with statements and questions that could not be recognised by the main argument parser. This provided a considerable linguistic robustness of the system. The pattern matching question answering interface does not have any theoretical value, and hence it was excluded from the evaluation.

The sentence "Go forward slowly" in (2) is an instruction of motion for the robot (Section 4.5.5.1). Unfortunately, we were unable to test the live performance of the sys-

tem on these commands with human evaluators. This was because it was considered too dangerous to control the robot purely by linguistic knowledge that was obtained automatically in a setting where not enough secondary control of the robot's motion could be ensured. The environment in which the robot was moving contained objects and thus its space was limited. The system did not contain an obstacle detection system that would stop the robot in case it was approaching an object. Obstacle detecting must be implemented at the level of navigation either by relying on the laser data, or even better, the data from sonars. Since a considerable development time would be involved in designing such system, and because its design would go beyond the scope of this work, it was decided not to go on with the implementation. The system does contain a very crude control of motion. The robot stops after three seconds. This was considered too restrictive for the evaluation where participants were asked to concentrate on the naturalness of the system. It did nonetheless allowed us to evaluate the system qualitatively and the results were encouraging.

To answer questions such as "Where is the table?" (Type A) in (3) the system chooses another reference object at random and uses a classifier to predict the best relation between the locations of these objects (Section 4.5.5.2). This setting is identical to *pDescriber*. It is a general case and it will tell us how good the system is in predicting the relation and locating objects. It will also allow us to make comparisons between the other settings of *pDialogue* and *pDescriber*.

Questions such as "Is the table to the left of the chair?" (Type B) in (4) are answered by taking the locations of both objects and using a classifier to find the best description of relation between them. If this is the same as the relation from the question, the answer is "yes", otherwise it is "no" (Section 4.5.5.3). In the evaluation we placed the robot and the objects so that the answer would always be "yes" according to our intuition. Assuming that the classifier is a perfect predictor (which is not), the setting gives us an indication how good the system is in finding the best description of relation

between the two objects.

Questions such as "What is to the left of the chair?" (Type C) in (5) are answered by finding a square region that corresponds "to the left of" the location of the reference object "the chair". The system returns a list of objects found in that region (Section 4.5.5.4). Assuming that the underlying classifiers are perfect predictors, the setting should give us an indication how good is the approximation of space into square regions for different descriptions of object relations.

Finally, questions such as "What is the chair to the left of?" (Type D) in (6) are answered by returning a set of possible reference object candidates that fall within a certain square region given that we know the location of the located object and the relation (Section 4.5.5.5). Assuming that the underlying classifiers are perfect predictors, the test should give us an indication whether the space for reference objects can also be approximated by a square region and what effect properties of reference objects have on their selection given that the system is not discriminating between them.

5.3.2 Experiment design

pDialogue was evaluated by a group of 13 volunteer subjects identified with letters from *a* to *m*, thus more than twice the number of subjects that evaluated *pDescriber*. Most of them were new to the experiment. Only three of them have already participated in the evaluation of *pDescriber*, and only two of them have participated in the experiment where data was collected for machine learning. There was a half-half distribution of native and non-native but fluent speakers of English.

The pre-experiment procedure was almost identical to that of *pDescriber* (Section 5.2.1). Objects were placed at various locations in the experiment room. Their placement was different from the placements in the previous experiments. A SLAM map was built and the objects were manually grounded on the map. The maximum size of the room and the maximum speed were found. These are used by *pDialogue* to normalise its observations which are subsequently applied to its classifiers. The classifiers that were used

were built by the J48 learner on the time-shifted *All* dataset. Numeric target classes were discretised to 11 nominal classes or bins. Each participant was first familiarised with the environment and the names of the objects. They were informed that the purpose of the experiment is to test the quality of the robot's linguistic response to questions about the location of objects in the room. Hence, their task was to judge whether the robot's answers are good and intuitive descriptions of the scene that the robot is currently in. A distinction how descriptions can be made relative to different reference frames was exemplified to them and they were asked to consider the descriptions as being made by the robot from its own perspective.

Although the robot can answer any question that falls within its knowledge base at any location, to control the experiment a script of questions was devised that were asked by every evaluator at the same predefined locations. To speed up the collection of data, the evaluators did not have to ask these questions themselves, but they were automatically submitted to *pDialogue* by *pDialogueEval*, a MOOS application which we specifically wrote for this purpose. The operator first placed the robot in one of the locations. When the evaluator was ready, they first heard a question and then an answer, both pronounced by the speech synthesiser. They were reminded that they should consider the question as being asked by them, whereas the answer is returned by the robot. After each question-answer pair, the evaluator, sitting behind a computer terminal, was prompted to evaluate the answer on a scale ranging from 1 to 5 where 1 is completely unsuitable, 3 is acceptable but probably there is another better description, and 5 is a perfect description. After choosing one of these values the evaluators also had a chance to type in comments about the description or their evaluation that they considered relevant, for example "ok under the perspective of the other object, did not include the chair". This allowed us to collect some qualitative data. Each run of the experiment took from 45 minutes to an hour to complete.

Figure 5.16 shows the SLAM representation of the environment including the names

of the objects. The map also shows the four locations marked as Location 1, 2, 3 and 4 in which the robot was placed when the questions were evaluated.

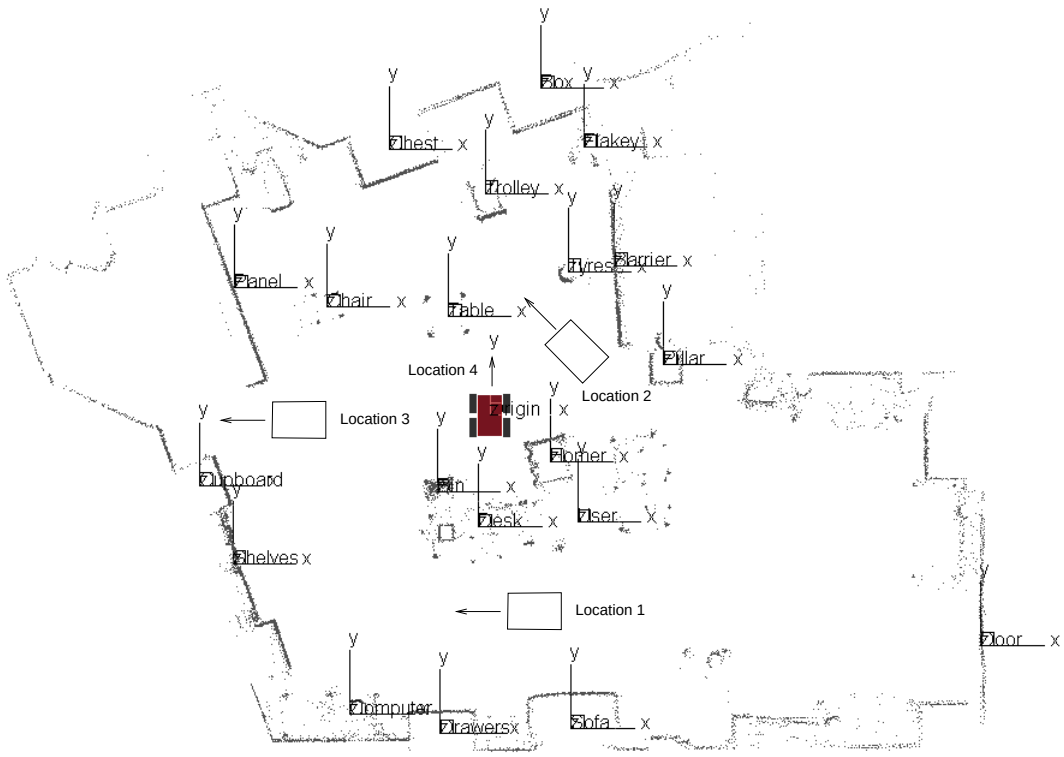


Figure 5.16: The evaluation environment for *pDialogue*

Table 5.11 shows the questions that were asked at these locations including their question types. For Location 1 we designed the questions so that the reference object is always the robot. We did not include question types A and D ("Where are you?" and "What are you to the left of?") because here the robot is the located object. Because the location of the robot is at stake it cannot fix the orientation of the reference frame. In such cases this must be fixed by the reference object which means that it can only be an intrinsic reference frame. Such configurations were excluded from learning (see Section 2.4.2.2, page 45).

Location 2 concentrates on objects that are in the robot's "field of vision". Question type A queries the robot about the location of four such objects which are at different distances away from it. The system returns the best description of the relation between

that object and another randomly chosen object which may not be in the robot's vision field. Question types B, C and D concentrate on the four most frequent descriptions of relation "to the left of", "to the right of", "in front of" and "behind". In question type B, both the reference and the located object are fixed by the question; in question type C the reference object is fixed; and in question type D the located object is fixed.

Location 3 concentrates on objects that are not in the robot's field of vision. The objects chosen for the answers are not restricted. Otherwise, the nature of questions is identical to those used in Location 2.

In Location 4 we test the three non-projective descriptions of object relations: "near", "close" and "far from". For this reason we exclude the question type A. For each relation the question types B and C were asked twice: once where both objects (B) or the located object (C) were in the "field of vision" of the robot and one when they were not. The question type D was only asked with a located object in the robot's "field of vision" because it is not natural to form this question with objects that are not.

Scene		Question	Type
Location 1	1.	Is the sofa to the left of you?	B
	2.	Is the desk to the right of you?	B
	3.	Are the shelves in front of you?	B
	4.	Is the door behind you?	B
	5.	What is to the left of you?	C
	6.	What is to the right of you?	C
	7.	What is in front of you?	C
	8.	What is behind you?	C
Location 2	1.	Where is the table?	A
	2.	Where is the chest?	A
	3.	Where is the barrier?	A
	4.	Where is the panel?	A
	5.	Is the chair to the left of the trolley?	B
	6.	Are the tyres to the right of the table?	B
	7.	Is the trolley in front of the chest?	B
	8.	Is the trolley behind the tyres?	B
	9.	What is to the left of the barrier?	C
	10.	What is to the right of the chair?	C
	11.	What is in front of the chest?	C
	12.	What is behind the table?	C
	13.	What is the chair to the left of?	D

Scene		Question	Type
	14.	What is Flakey to the right of?	D
	15.	What are the tyres in front of?	D
	16.	What is the chest behind of?	D
Location 3	1.	Where is the bin?	A
	2.	Where is the pillar?	A
	3.	Where is the table?	A
	4.	Where are the drawers?	A
	5.	Is the user to the left of Homer?	B
	6.	Is the origin to the right of the desk?	B
	7.	Is the bin in front of Homer?	B
	8.	Is the user behind the bin?	B
	9.	What is to the left of the table?	C
	10.	What is to the right of the desk?	C
	11.	What is in front of the pillar?	C
	12.	What is behind the table?	C
	13.	What is Homer to the left of?	D
	14.	What is the table to the right of?	D
	15.	What is the bin in front of?	D
	16.	What is the pillar behind of?	D
Location 4	1.	Is the table near the origin?	B
	2.	Are the drawers near the sofa?	B
	3.	Is the barrier close to the tyres?	B
	4.	Is the origin close to the user?	B
	5.	Is Flakey far from the chair?	B
	6.	Is the computer far from Homer?	B
	7.	What is near the trolley?	C
	8.	What is near the desk?	C
	9.	What is close to the chair?	C
	10.	What is close to the user?	C
	11.	What is far from the barrier?	C
	12.	What is far from the sofa?	C
	13.	What is the trolley near to?	D
	14.	What is the chair close to?	D
	15.	What is the barrier far from?	D

Table 5.11: Questions that were asked at each location

5.3.3 Evaluator agreement

Parallel to the evaluation of *pDescriber* (Section 5.2.2) we need to establish whether the evaluators behaved similarly while evaluating the performance of *pDialogue*. There

we used statistical correlation to estimate the consistency of evaluator behaviour. We also use this method here. We introduced two related linear correlation coefficients: Pearson's product moment coefficient (r_{xy}) and Spearman's coefficient of rank correlation (ρ). We calculated them between a set of values from one evaluator and the mean of the values from the rest of the group. The set of items that the values represent must be identical across all evaluators. The procedure was repeated for each remaining evaluator. The mean of the values obtained through each fold was considered as an estimation of the overall evaluator agreement.

Evaluators behave similarly if they evaluate responses of the system to questions in the same way. Since the questions that were answered are the same for evaluators, unlike with *pDescriber*, the individual judgements of evaluators can be compared directly. It is important to note that the response of the system may not always be the same and not much more can be done to ensure its consistent behaviour. This adds adds noise to the estimation. Each judgement was a value on a 1 to 5 ordinal scale. The correlation can be calculated using either r_{xy} or ρ. In the latter case, the correlation is calculated not from the actual scores, but from the rank that is assigned to each question depending on the magnitude of its score in comparison to the scores of other questions. Table 5.12 shows the resulting coefficients for each fold and their overall means and Figure 5.17 represents them graphically. The number of items correlated in each fold is 55 which corresponds to the number of questions.

As before, the *s in Table 5.12 indicate the level of statistical significance of the coefficients obtained by a two-tailed t-test. * indicates that the correlation is significant at the 0.05 level, ** indicate that it is significant at the 0.01 level and "ns" indicates that it is not statistically significant. The table shows that the majority of the correlation coefficients are significant at the level $\alpha = 2P = 0.01$ which means that the risk of rejecting the zero hypothesis which states that there is no correlation between the scores is less than 1% (Type I error). Both correlation coefficients for the evaluator j are statis-

Fold	r_{xy}	ρ
a:rest	0.617**	0.639**
b:rest	0.398**	0.395**
c:rest	0.613**	0.559**
d:rest	0.591**	0.567**
e:rest	0.719**	0.709**
f:rest	0.770**	0.734**
g:rest	0.263ns	0.389**
h:rest	0.702**	0.708**
i:rest	0.534**	0.537**
j:rest	0.297*	0.321*
k:rest	0.638**	0.604**
l:rest	0.755**	0.779**
m:rest	0.686**	0.674**
Mean	0.583	0.586

Table 5.12: Evaluator agreement per fold

tically significant at the 0.05 level, and only one coefficient, the r_{xy} for the evaluator g is statistically non-significant.

The t-test confirms that the correlation coefficients are representative measures of agreement between the evaluators. Now let us turn to their values. Table 5.12 and Figure 5.17 show a similar situation that was observed when agreement was estimated for *pDescriber*. The correlation between the majority of individual evaluators and the rest of the group is medium ($0.40 < r \leq 0.70$, 6 cases) to high ($0.70 < r \leq 0.90$, 4 cases) which indicates a good measure of agreement.[6] There are however three cases (b:rest, g:rest and j:rest) where the correlation coefficients are lower than or equal to 0.40 and here we can only speak of a low positive correlation ($0.20 < r \leq 0.40$). It appears again that a minority of evaluators (23% here and 20% in the evaluation of *pDescriber*) behave differently from the rest of the group. As before, it is difficult to make any firm conclusions why this is so. The agreement between the 10 evaluators is encouraging and shows that the system performs uniformly across different human observers. The

[6]These descriptive categories are based on observations and experience rather than scientific facts as outlined in (Sagadin, 2003, page 122).

remaining three evaluators may have approached the task with different expectations and consequently their scores are different. The mean correlation coefficients r_{xy} and ρ are 0.583 and 0.586 respectively.

Fold	r_{xy}	ρ	r_{xy}	ρ
	Group A		Group B	
a:rest	0.642**	0.664**		
b:rest			0.299*↓	0.289*↓
c:rest	0.581**↓	0.537**↓		
d:rest	0.639**	0.628**		
e:rest	0.751**	0.746**		
f:rest	0.803**	0.769**		
g:rest			0.228↓	0.249↓
h:rest	0.674**↓	0.668**↓		
i:rest	0.540**	0.521**↓		
j:rest			0.251↓	0.353**
k:rest	0.608**↓	0.580**↓		
l:rest	0.746**↓	0.763**↓		
m:rest	0.713**	0.686**		
Mean	0.670	0.656	0.259	0.297

Table 5.13: Evaluator agreement per fold for two evaluator groups

To further exemplify the differences between the two groups of evaluators, we separated them into two sets $A = \{a, c, d, e, f, h, i, k, l, m\}$ and $B = \{b, g, j\}$ and recalculated the coefficients per folds as shown in Table 5.13. We expected that excluding evaluators b, g and j would result in improved correlation coefficients. Although the means of the coefficients r_{xy} and ρ increased to 0.670 and 0.656 respectively, as shown in the left part of Table 5.13, this does not hold for each coefficient from each fold. The coefficients that decreased in value are marked with $\downarrow$ and they represent almost a half of the coefficients in that part of the table (9/20). We also expected that creating a group of evaluators b, g and j would result in an increased agreement within this group. This assumption has been disproved as the majority of their coefficients (5/6) decreased as shown in the right part of Table 5.13. The coefficient values therefore suggest that it is impossible to conclude that there are two groups of evaluators. This means that evaluators b, g and j cannot be taken as "exceptional cases" since their agreement was not

uniformly different from the agreement of other evaluators.

This is to be expected. The scores that were correlated were evaluations of different tasks/question types and evaluators could have had different bias for each of them which has a negative effect on the correlation score. This bias was not present in the estimation of agreement between the evaluators of *pDescriber* because there the evaluated descriptions were all of the same type. Furthermore, the bias could also be due to the system bias on particular evaluation instances. As discussed previously, not all tasks were the same in terms of the decisions that the system made and also there was no guarantee that the system responded exactly the same to each task each time. We did the best we could to keep the conditions for each evaluation run identical, but they were not completely identical. Overall, we can conclude that there is considerable agreement between the evaluators which has been estimated as a medium positive correlation and that the result is good given the nature of evaluation.

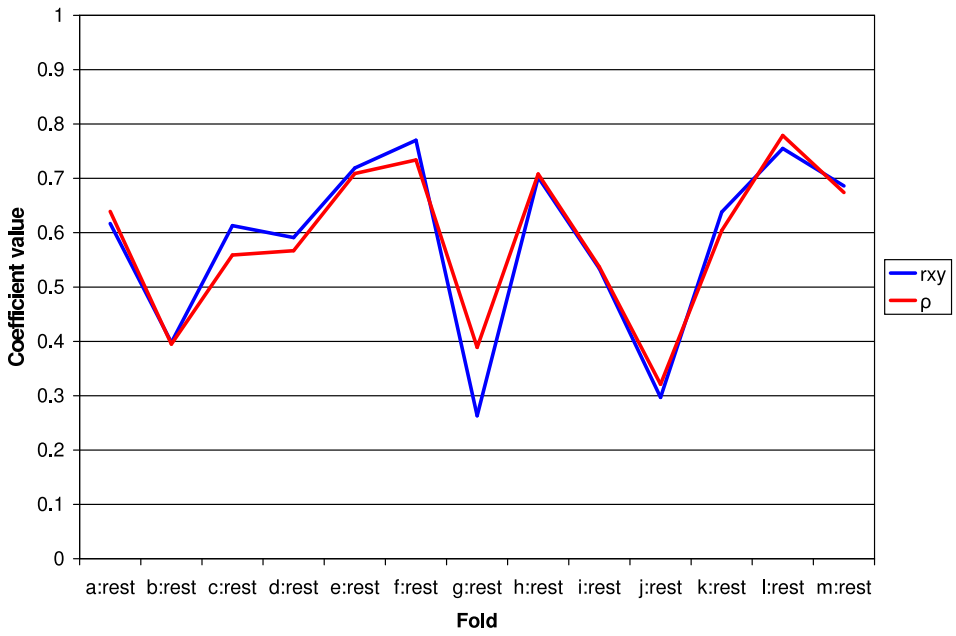


Figure 5.17: Evaluator agreement per fold

Before concluding this section let us briefly discuss the difference between the two correlation coefficients r_{xy} and ρ. In Figure 5.17 we can see that their values are surpris-

ingly similar, if not even the same. Why is this so? When we introduced the coefficients in Section 5.2.2, we explained that Person's ρ is calculated using the same formula as Spearman's r_{xy}, the only difference is that it is calculated from ranks rather than interval or ratio data. The coefficient values are similar because the source data from this evaluation experiment already reflects the ranks. Each item from the first set of scores holds an integer value between 1 and 5. When such items are ranked in the calculation of ρ, the relations between the original values are preserved. For example, suppose that an evaluator assigned the score 2 to 4 questions from the list and these questions should occupy places 3, 4, 5 and 6 on the rank scale. Because the items hold the same score, they are assigned the same tied rank 4.5 ($(3+4+5+6)/4$). The same holds for the second set of scores. Most likely each mean will be a unique value. If so, it will be assigned a unique rank. In both cases the relation between the original scores is preserved when these are translated to ranks. The coefficients are similar because both sets of scores are used in the same equation. They are not completely identical, because the mapping to ranks within one set of scores does not correspond to a linear function and because the assignment of (tied) ranks depends on the distribution of identical values within a set of scores.

5.3.4 System performance

The evaluators were asked to provide judgements of the system's response to four question types that the system answered at four different locations. Each question type uses different classifiers in different configurations. The questions made at these locations distinguished between objects that were in the "vision field" of the robot (Location 1 and 2) and those that were not (Location 3). Location 1 differs from Location 2 in questions which always contain the robot as the reference object and thus enforce the usage of the intrinsic reference frame fixed on the robot. We assumed that this will help evaluators later to choose the relative reference frame which is also fixed by the

orientation of the robot. The choice excludes the question types A and D from Location 1. We made a distinction between projective prepositions (Locations 1, 2 and 3) and non-projective prepositions (Location 4). Therefore, the experimental settings reflect a considerable variation of conditions.

In this section we examine the overall performance of the system which we define as the ability of the system to generate linguistic descriptions that appear natural to a human observer as if they were made by another human. Since the performance is a highly subjective measure we also estimated the agreement between the group of evaluators. The evaluators judged the responses of the system on a scale from 1 to 5, where 5 meant the highest and 1 the lowest score. However, here we want to measure the performance of the system in percentages as we did in the evaluation of *pDescriber* in Section 5.2.3. This can be accomplished by weighting each each 1 by 0, each 2 by 0.25, each 3 by 0.50, each 4 by 0.75 and each 5 by 1. The performance can be expressed as the sum of the weighted scores over the total number of scores in a particular subset. The evaluation scores for *pDialogue* are thus comparable to the binary evaluation scores in the evaluation of *pDescriber*, but because they are graded, they allow finer distinctions.

Table 5.14 shows the scores per question type for all locations as evaluated by 13 evaluators compared with the accuracies of the underlying classifiers that were used in each case. In the case of questions C and D two classifiers are used. The classifier accuracies were taken from Table 3.7, page 105 for *All-J48* and Table 3.13, page 124 for *All-J48-11bins*. The estimated *pDialogue*'s performance is in most cases lower than the accuracy of the underlying classifiers that the system was using. The only exception is the performance value for the question type C (54.70%) which is higher than the accuracy of one of the underlying classifiers (48.80%). However, it is nonetheless markedly lower than the accuracy of its second classifier (72.80%).

In Table 5.8 we cited that the estimated performance of *pDescriber* on the *Relation* category is 59.28%. As mentioned before, finding an answer to question type A corre-

Question type	Accuracy (%)	Classifier	Accuracy (%)
A	43.51	relation	69.12
B	54.17	relation	69.12
C	54.70	lo_x	48.80
		lo_y	72.80
D	56.92	refo_x	65.60
		refo_y	82.24
Mean	52.33		67.71

Table 5.14: System performance and classifier accuracy

sponds closely to generating a description in *pDescriber* in terms of the choices that the system makes. Furthermore, both *pDescriber* and *pDialogue* use the same underlying classifier to predict the relation which has an estimated accuracy of 69.12%. In spite of these similarities the estimated performance of *pDialogue* on question type A (43.51%) is lower by 15.69% from the estimated performance of *pDescriber*.

One reason for the difference is that both systems have not been evaluated with the same evaluation metric. In *pDescriber* evaluators were only queried whether descriptions were acceptable or non-acceptable, wheres a five point scale in the evaluation of *pDialogue* allowed them to make finer distinctions. Our weighting of the five-point scale is conservative compared to the binary tags that discriminate only between "acceptable" and "unacceptable" descriptions. The scale distinguishes between three different levels of acceptability which are marked by the scores 3, 4 and 5. Under the binary scheme the score 3 would most likely count as an acceptable case, whereas weighting it by 0.5 makes it neither "acceptable" nor "unacceptable". Indeed, if we simply count all 1s and 2s unacceptable and all 3s, 4s and 5s as acceptable, the estimated system performance of *pDialogue* on the answers to question A improves to 50.96%. This demonstrates the effect of different metrics on the evaluations.

Another reason for the lower performance of *pDialogue* is that generating a description and answering a question are perhaps not the same. For example, if the system

generates a description on its own, a human hearer understands it as a statement about the scene that both participants are observing. On the other hand, if a human is asking the system about the location of some object, the answer must be informative or relevant to the hearer so that they can locate the object in the scene. Choosing a salient reference object is particularly important. Since *pDialogue* does not implement a model of salience, it under-performs in this category of descriptions.

Let us now turn to the performance of the system answering questions of the type B. Table 5.14 shows that the overall performance of the system on this question type is 54.17% which is comparable to its performance on other questions. However, the evaluations of the question type B ("Is the desk to the right of you?") can be separated into two sub-cases. The system answers these questions by using the classifier to predict a relation between the specified located object and the reference object. If the predicted relation is the same as the relation in the question, the system answers "Yes - the desk is to the right of me", otherwise it answers "No - the desk is behind me", thus reporting the relation predicted by the classifier. For the experiment we chose such questions so that in our intuition the answer would always be "yes". We also considered a reference to objects that were not in the "vision field" of the robot as acceptable.

Answer	Location	n	Performance (%)
yes	all	67	93.28
yes	1, 2	43	98.84
yes	3	24	83.33
yes	4	none	none
no	all	165	38.64
no	1, 2	60	20.83
no	3	27	37.04
no	4	78	52.88

Table 5.15: *pDialogue*'s performance answering question type B

The first part of Table 5.15 shows cases where the author of the experiment and the system agree on the choice of the best relation. It can be seen that in this case the performance of the system is estimated high - 93.28% in overall. Further it can be seen

that evaluators agree almost perfectly with the system and its author on the choice of the best description in Locations 1 and 2 where the objects were in the "vision field" of the robot, whereas there is a decrease in the estimated performance for Location 3 (83.33%) where the objects were not in the "vision field" of the robot. This indicates that some evaluators dislike descriptions referring to objects that are not "visible" to the robot. We return to this question in the following section.

In Location 4 we were dealing with non-projective/topological descriptions of object relations such as "close to" and "far from". There were no cases where the system would choose the same relation as the author of the experiment and hence there are no evaluations in this section. This is because topological descriptions compete with projective descriptions and according to the classifiers the latter are the preferred ones to describe the location of objects. A possible reason for this is that projective descriptions allow a more precise grounding of objects. They include proximity between two objects and the orientation of the reference frame rather than just the former. Because our system treats both kinds of object relations in the same way, topological descriptions are less likely to be generated, if at all. This suggests that each type of descriptions should be learned separately and that an improved system should choose between the two kinds of object relations depending on the level of detail that is required for each generation.

The author of the experiment and the system chose the same best description in 28.88% (67/232) of cases. Table 5.8 shows that the average estimated performance of *pDescriber* on the *Relation* category across 5 evaluators is 59.28%. The scores range from 43.10% to 72.73%. This is because a certain scene can be described with more than one description of relation. When generating an answer to a question type B *pDialogue* behaves in a more restricted way than humans did when they evaluated *pDescriber*. The humans were allowed to hear the robot's reply before they supplied their own judgements, whereas *pDialogue* only relies on its classifiers which predict a

single relation. The figure 28.88% is an estimation from a single evaluator and cannot be taken as representative of the system's performance. It does however positively tell us that the system has internalised a considerable degree of human preference for a single best referring expression of relation for a particular configuration of objects that maximally distinguishes them from other object configurations.

Most likely, the performance of the system would improve if it considered more than just one best description. This is confirmed by the figures in the second part of Table 5.15, by the cases where the system chose a different description of relation than the author of the experiment. The evaluators most often agree with the alternative description made by the system in Location 4 (52.88%). This is expected. As mentioned previously the questions asked at this location contain topological descriptions of relations that compete with projective descriptions which are also the preferred ones. The questions for Locations 1, 2 and 3 only contain projective descriptions. Although the evaluators show a general tendency to dislike the descriptions made by the system and thus agree with the author of the experiment, they also show a considerable preference – 20.83% and 37.04% respectively – for these alternatives. Furthermore, these figures are pessimistic estimations as they may have been decreased by the genuine classifier errors.

To answer the questions type A and B the classifier predicts a linguistic description which is a discrete symbolic class. However, for the question type C ("What is to the left of the barrier?") and the question type D ("What is the chair to the left of?") the classifiers predict a region in space. Subsequently, the system finds a set of objects that belong to that region and generates an answer. In Section 3.3.4, page 74 we mentioned that the chosen classifiers can only predict nominal categories of target concepts and not values on a continuous numeric scale. The latter define the location of objects. To use the location of an object as a target concept, we discretised the environment into nominal regions each representing an interval. The number of intervals that we chose

influenced the measured accuracy of the classifier (Section 3.5.5). The fewer intervals we chose, the more accurate was the classifier. It meant that there was less probability that an instance would be assigned to an incorrect class. However, the number of intervals also influences to what detail the system is able to discriminate regions of space. For example, the system is able to generate descriptions of objects that are more meaningful and precise to humans if the space is partitioned into five regions rather than just two. The number of partitions must be chosen carefully to balance between both factors.

For the classifiers used in the evaluation of *pDialogue* we chose 11 bins as the optimal number based on their performance. As shown in the right part of Table 5.14 the accuracies of the classifiers whose target concepts were discretised (*LO_x*, *LO_y*, *REFO_x*, *REFO_y*) are comparable with the accuracies of those classifiers whose target concepts were genuinely nominal (*Relation*). However, this trend must also be replicated in the system performance evaluated by humans. The left part of Table 5.14 shows encouraging performance figures on the question type C and the question type D: 54.70% and 56.92% respectively. They are higher than the performance of the system on the question type A where the classifier predicts true nominal categories (43.51%) but the reason for this may be that answering a question A also requires object selection and not just the application of the classifiers. The performance figures on question type C and D also approach the value estimated for *pDescriber* (59.28%). Since both the classifier accuracy and the system performance are balanced, we can be sure that choosing 11 bins in the discretisation step was a good choice.

The preceding discussion confirms that overall the system performs well and can replicate a considerable amount of human competence when generating descriptions. The performance figure of 50% translates to the evaluation score 3 which indicates an acceptable performance with a possibility of improvement. Considering that the performance metric used in this evaluation is more conservative than the one used for

the evaluation of *pDescriber* and that the evaluated items were selected to represent a cross-section of (extreme) cases that the system should deal with, the results are indeed positive and may be comparable to those from the evaluation of *pDescriber*. Discussing the system's performance on question type B we have seen that there may be considerable differences in performance when different sets of conditions, such as the locations and their associated question types, applied. The purpose of the evaluation was to show under which conditions the system under-performs. We examine these differences in performance in the following section.

5.3.5 Differences in performance

We demonstrate the difference in performance of *pDialogue* under different sets of conditions with two statistical tests.

5.3.5.1 t-test

In Section 5.2.2, page 188 we discussed how the t-statistic can be used to determine whether a sample correlation coefficient is indicative of the correlation coefficient of the population. In a slightly more complicated scenario, the t-test can be used to determine whether two populations, in our case sets of scores, differ in the arithmetic means of their items (Sagadin, 2003, Chapter 12, page 214ff.). Without performing the test, this could only be established by examining each item of the population and calculating the means. This would be quite time consuming or even impossible since the number of evaluations corresponds to the number of descriptions that can be made which is potentially infinite. Instead, we collected samples of evaluations from which we can determine sample means. Using the t-test we can determine if the differences in sample means are indicative of the differences in population means.

The zero hypothesis for the t-test H_0 states that there is no difference in the means of the two populations or that $M_{x1} - M_{x2} = 0$ or $M_{x1} = M_{x2}$. In contrast, the alternative

hypothesis H_1 states that the means are different, $M_{x1} \neq M_{x2}$, or that $M_{x1} - M_{x2} \neq 0$. The test is two sided.

To calculate the t-statistic (Sagadin, 2003, Equation 225, page 229) we need to know the size of the two samples (n_1 and n_2), their means ($\bar{x}_1$ and $\bar{x}_2$) and their variances (s_1^2 and s_2^2). The test makes two assumptions which must hold for it to be valid. The variables x and y must be distributed normally in both populations and with identical variance σ^2 ($\sigma_1^2 = \sigma_2^2 = \sigma^2$). The latter can be confirmed with the Levene's test for equality of variances which was introduced by Levene (1960). The test calculates another statistic known as the F-statistic (Sagadin, 2003, page 241ff.).[7] Because the t-test requires a confirmation that the variances of both populations are equal, the zero hypothesis H_0 must be confirmed rather than rejected in this case.

The values of t are distributed in a probability distribution known as the t-distribution with $g = n_1 + n_2 - 2$ degrees of freedom. This means that a value of t falls within the interval $-t_P$ to $+t_P$ (for $g = n_1 + n_2 - 2$ degrees of freedom) with a probability $1 - \alpha$ and it falls outside this interval with a probability $\alpha = 2P$. The most commonly chosen critical value P for two-tailed statistical tests is 0.025. If the absolute value of the calculated t is greater than or equal to $t_{P=0.025}$, the probability that the zero hypothesis is true is very small, $\alpha = 2P \leq 0.05$. We can reject such a hypothesis, the risk that the rejection is false is $\leq 5\%$. If the zero hypothesis is rejected, the alternative hypothesis is accepted. The risk that the alternative hypothesis is false is $\leq 5\%$. Therefore, when reporting results we say that if $|t| \geq t_{P=0.025}$ for $g = n_1 + n_2 - 2$ degrees of freedom, the difference of sample arithmetic means is statistically significant at the level of risk $\alpha = 2P \leq 0.05$.

[7] The F-statistic distributes according to the F-distribution with $g_1 = 1$ and $g_2 = n_1 + n_2 - 2$ degrees of freedom.

5.3.5.2 Pearson's Chi-square test

The t-test compares the means of two samples of an interval or ratio variable if they were drawn from the same or different populations. On the other hand, the Chi-square (χ^2) test (Sagadin, 2003, Chapters 17–19, page 293ff.) compares the frequencies of values of two nominal variables in a sample. The frequencies are obtained by cross-tabulating the values of the nominal variables in a contingency table. Each cell of a contingency table tells us the number of items that had some value of the first variable and some value of the second variable. For example, one of the questions that we would like to answer is whether evaluator scores collected in locations where the objects referred to are in the "vision field" of the robot (Location 1 and 2) are different from the scores collected in locations where the objects are not in the "vision field" of the robot (Location 3). These values would be represented as in Table 5.16.

Location/Score	1	2	3	4	5	Total
1&2	31	22	34	53	68	208
3	39	30	27	27	33	156
Total	70	52	61	80	101	364

Table 5.16: The scores for Location 1&2 against the scores for Location 3 for a sample of 364 evaluations

The comparison of frequencies can be made in terms of goodness of fit or to test their independence. The zero hypothesis (H_0) in the χ^2 test states that there is no dependency between the frequencies of values of the two variables. In our example, this would state that there is no difference in scores between the two types of locations. If H_0 is true, then we expect that the distribution of nominal values is the same with both variables.

For example, the score 5 is assigned to 101 out of 364 evaluations in Table 5.16. Converted to a ratio this equals to $101/364 = 0.2774$. If the zero hypothesis is true, than the same ratio of the score 5 should be found with Location 1&2 and 3. However, this is not the case, the ratios are $68/208 = 0.3269$ and $33/156 = 0.2115$. Using the first

ratio we can calculate the expected or theoretical frequency for each cell of Table 5.16. For example, the observed frequency of the cell (1&2, 5) is 68, the expected frequency is

$$\frac{f_t}{208} = \frac{101}{364} \tag{1}$$

$$f_t = \frac{101 \times 208}{364} = 57.7143 \tag{2}$$

The observed frequency for the cell (1&2, 5) is higher than if there were no dependency between the variables ($68 > 57.71$). We can test whether the sample difference between the observed and the expected frequencies is statistically significant in respect to the underlying population by calculating the χ^2 statistic (Sagadin, 2003, Equation 323, page 296). If H_0 is true, then the calculated χ^2 distributes in a χ^2-distribution with $g = (c-1) \times (r-1)$ degrees of freedom.[8] H_0 is rejected for $\alpha = P \leq 0.05$.

To summarise, both statistical tests, the t-test and the χ^2 test, allow us to make inferences in the form of statistical significance about the distribution of scores in our evaluation. They examine the scores in a slightly different way and this is also the reason why we analyse our data with both. The t-test assumes that scores are interval or ratio variables and compares the means of these variables between the two samples, for example the two locations. The χ^2 test assumes that scores are categories and compares how these are distributed between two variables, in our case two locations. The test determines whether the observed and predicted frequencies of categories are taken from the same population or not. The χ^2 test is a more appropriate statistical test if we assume that the differences between individual scores are not always equal on a numeric scale.

[8] c stands for the number of columns and r for the number of rows in the contingency table.

5.3.5.3 The effect of the field of vision

Many evaluators of *pDescriber* pointed out that they dislike descriptions that are referring to objects not in the "field of vision" of the robot and they considered the system to over-generate (Section 5.2.5.4). A comparison of the system's performance in Locations 1&2 where the objects were "visible" to the robot and Location 3 where the objects were behind the robot's "back" shows that the descriptions of the latter are indeed evaluated lower (Table 5.17). The only exception are the answers to question type B which we already discussed on page 229. The statistical tests will show whether the evaluator scores in Table 5.17 are different between the two configurations in general and not just in the sample. To exclude the influence of other factors we limit our samples to Locations 1&2 and 3 and question types A, C and D.

Location/Question type	A	B	C	D	All
1&2	45.19	53.13	63.22	78.85	59.46
3	41.83	58.17	45.67	55.29	50.24

Table 5.17: System performance (in %) per question type on descriptions of "visible" (Location 1&2) and "non-visible" objects (Location 3)

We first consider the t-test. The size of the sample for Location 1&2 is $n_1 = 208$ and for Location 3 this is $n_2 = 156$. Their corresponding means are $\bar{x}_1 = 3.50$ and $\bar{x}_2 = 2.90$ and their variances are $s_1 = 2.019$ and $s_2 = 2.217$. In order for the t-test to be valid, the Levene's test must be applied first to show that the population variances are equal ($\sigma_1^2 = \sigma_2^2$). The calculated F statistic for the samples is 0.852 which is less than the critical value of $F_{P=0.05}$ ($g_1 = 1$, $g_2 = 362$) which lies approximately between 3.86 ($g_1 = 1$, $g_2 = 400$) and 3.89 ($g_1 = 1$, $g_2 = 200$). Therefore, we accept the zero hypothesis on the equality of population variances. The risk of rejecting it is too high $\alpha = P = 0.357$.[9]

[9]The precise value of P is calculated by the SPSS statistical package. This cannot be determined from the table of scores of the F-distribution which is standardly used in the manual calculation of the F-statistic.

The t statistic is $t = 3.912$. This is greater than the critical value $t_{2P=0.001}$ ($g = 362$) which is approximately 3.32 ($g = 300$ and $g = 400$). Therefore, the zero hypothesis on the equality of population means can be rejected with a risk $\alpha = 2P = 0.000 < 0.001$. We accept the alternative hypothesis which states that the population means are not equal.

Because $\bar{x}_1 > \bar{x}_2$ it also follows that $M_{x1} > M_{x2}$. We can estimate the difference between the population means as a confidence interval with some risk $\alpha = 2P$ (Sagadin, 2003, Equation 257, page 245). The estimated lower and upper bound of this interval are 0.299 and 0.903 ($\alpha = 2P = 0.05$). This means that in general the mean of scores of descriptions of objects that are not in the "vision field" of the robot is with a likelihood of 95% lower by 0.299 to 0.903 from the mean of scores of descriptions of objects that are in the "vision field" of the robot. This is considerable considering that the scores range from 1 to 5.

Let us now test the same dataset with the χ^2 test. Its contingency table has already been given in Table 5.16. One of the conditions for this test is that a theoretical frequency of a cell is not less than 5. This condition is satisfied. The value of χ^2 is 16.434. This is more than the critical value of $\chi^2_{P=0.01}$ ($g = 4$) = 13.28 and less than the critical value of $\chi^2_{P=0.001}$ ($g = 4$) = 18.47. The zero hypothesis that there is no difference in the frequencies of scores between Location 1&2 and Location 3 can be thus rejected with a risk $\alpha = P = 0.002 \leq 0.01$.

The χ^2 test measures contingency or strength of association between two nominal variables which can be expressed as a contingency coefficient. One of the contingency coefficients based on χ^2 is Cramer's V (Sagadin, 2003, Equation 338, page 316). V expresses the association between two variables as a percentage of their maximum possible variation. A perfect relationship between the variables is the one which is predictive or ordered monotonically. A null relationship is statistical independence. The coefficient does not tell us the direction of association. This must be determined

separately from the frequencies of a contingency table. Table 5.16 gives us the value of V as 0.212. The relationship between the variables is most likely negative as the frequencies of low to high scores decrease when progressing from Location 1&2 to 3. Testing the zero hypothesis in the χ^2 test also tests the zero hypothesis that the population $V = 0$. The risk of rejecting both zero hypotheses is the same.

In this section we have shown that the difference in scores between Locations 1&2 and 3 is statistically significant at the level of risk $\alpha = 2P = 0.000$ (t-test) and $\alpha = P = 0.002$ (χ^2 test). In the case of the t-test we estimated with a confidence of 95% that in general the mean of scores for descriptions of objects not in the "vision field" of the robot is lower from the mean of scores for descriptions of objects in the "vision field" of the robot by 0.299 to 0.903. The χ^2 test allowed us to estimate the strength of association between the description scores and the location which is $V = 0.212$. Both tests indicate that the scores between Location1&2 and 3 are considerably different which means that the system should be adapted to avoid generating descriptions of objects that are not in the "vision field" of the robot.

5.3.5.4 Projective and topological relations

Projective relations such as "left" and behind' relate the objects alongside two dimensions: proximity and the orientation of the reference frame. Topological relations such as "near" and "far" only consider proximity between the objects. Our model for machine learning treated both in the same way which means that the orientation information was also encoded as an attribute when topological relations were learned. It was hoped that its effect would fade through the number of observations which would make the classifiers encode this property only in the most general way. During the evaluation experiment it became evident that considerable errors occurred on descriptions of topological relations because the classifiers also internalised the orientation component and therefore over-fitted on the training data.

In this section we examine if our impressionistic observations of evaluators are also reflected in their scores in general. The evaluators were given the task to evaluate topological relations in Location 4. There were no questions of type A in this location. Because question type B does not evaluate a single condition, we also exclude its scores from all locations. This leaves us with question types C and D which can be compared between Locations 1&2&3 where they were used with a projective relation and Location 4 where they were used with a topological one. We include Location 3 in the first set because Location 4 also contained descriptions of objects that were not in the "vision field" of the robot. In fact, question types C and D are also the most suitable for this test. In these cases in addition to the classification the system does not need to make any other decisions when generating answers. Secondly, they test the goodness of region predicted by the classifiers containing either potential LOs (C) or REFOs (D). The regions will directly reflect the effect of the orientation component on topological relations.

Table 5.18 shows the estimated system performance for the chosen configurations. The scores for Location 4 are lower than the scores for Location 1&2&3. The column "All" contains the estimated performance over all question types in these locations, thus also including question types A and B.

Location/Question type	C	D	All
1&2&3	57.37	67.07	55.77
4	49.36	16.35	46.15

Table 5.18: System performance (in %) per question type on descriptions with projective (Location 1&2&3) and topological relations (Location 4)

We consider the t-test first. The size of the sample Location 1&2&3 is $n_1 = 260$ and the size of Location 4 is $n_2 = 104$. Their means and variances are $\bar{x}_1 = 3.45$, $s_1 = 2.008$ and $\bar{x}_2 = 2.64$, $s_2 = 2.289$. The Levene's test of equality of population variances ($\sigma_1^2 = \sigma_2^2$) gives us the F statistic of 2.030. This is less than the critical value of $F_{P=0.05}$ ($g_1 = 1, g_2 = 362$) which lies approximately between 3.86 ($g_1 = 1, g_2 = 400$) and

3.89 ($g_1 = 1$, $g_2 = 200$). The zero hypothesis on the equality of population variances thus cannot be rejected ($\alpha = P = 0.155$). The t-statistic can be calculated which gives us the value of $t = 4.805$. This is greater than the critical value $t_{2P=0.001}$ ($g = 362$) = 3.32. The zero hypothesis that the population means M_{x1} and M_{x2} are equal can thus be rejected ($\alpha = 2P = 0.000$). Because $\bar{x}_1 > \bar{x}_2$ it also holds that $M_{x1} > M_{x2}$. With 95% confidence M_{x1} is higher than M_{x2} in the interval of 0.476 to 1.136.

Table 5.19 represents a contingency table for a sample of 364 evaluations for the χ^2 test. None of the theoretical frequencies are below 5 and hence the precondition for the test is met. The test gives us the value of $\chi^2 = 24.550$. This is more than the critical value of $\chi^2_{P=0.001}$ ($g = 4$) = 18.47. Therefore, we can reject the zero hypothesis and accept the alternative one that there is a difference between the scores between the two locations ($\alpha = P = 0.000$). The value of the Cramer's V coefficient of contingency is 0.260 and as exemplified by the scores in the table the association between the variables is a negative one.

Location/Score	1	2	3	4	5	Total
1&2&3	39	32	41	69	79	260
4	34	22	14	15	19	104
Total	73	54	55	84	98	364

Table 5.19: The scores for answers to question types C and D for Location 1&2&3 against Location 4 for a sample of 364 evaluations

Both tests confirm that the differences between two two sets of scores in terms of their population means and their frequency distributions are statistically significant at the level 0.000. The difference between the populations means is considerable. With 95% confidence this lies in the interval of 0.476 to 1.136. Thus, it can be concluded that including the orientation component in the machine learning data for topological relations has a negative effect on the performance of the system.

5.3.5.5 Choice of the reference frame

In all experiments we asked evaluators to describe the environment and evaluate the performance of the system from the perspective of the robot. This meant that they had to use intrinsic reference frame if the reference object was the robot and relative reference frame if the reference object was another object. The first case enforces the usage of the intrinsic reference frame. However, in the second case the reference frame can be changed between the intended relative frame and the intrinsic frame fixed by the reference object. It was noticed that with these descriptions both describers and evaluators sometimes changed the reference frame.

We can compare the scores obtained in the first configuration with the scores obtained in the second. The system cannot generate descriptions using an intrinsic reference frame fixed by a reference object other than the robot. Thus, if human observers prefer to change the reference frame, they would penalise the system for not doing so. The reason for the change is either because certain configurations and properties of objects preferentially select intrinsic reference frame or because the reference frame is flexible and the evaluators simply get confused. We test if such potential changes in the reference frame by the evaluators have a significant effect on the overall evaluation scores.

The robot is used as the reference object in question type C at Location 1. We can compare these scores with the scores for question type C at Location 2 where the reference object is different than the robot. The size of the samples are $n_1 = n_2 = 52$. Their corresponding means and variances are $\bar{x}_1 = 3.79$, $s_1^2 = 2.054$, $\bar{x}_2 = 3.27$, $s_2^2 = 1.847$. The usage of the t-test is justified by the Levene's test. The value of the calculated F statistic is 0.026 and the corresponding risk for rejecting the zero hypothesis on the equality of population variances for $g_1 = 1$, $g_2 = 102$ degrees of freedom is $\alpha = P = 0.873 > 0.05$. Therefore, the zero hypothesis cannot be rejected. The value of the t-statistic is 1.896. For this value of t and for $g = 102$ degrees of free-

dom the risk of rejecting the zero hypothesis on the equality of population means is $\alpha = 2P = 0.61 > 0.05$. The zero hypothesis cannot be rejected. With a confidence of 95% we expect to find the difference between the population means in the interval between -0.24 to 1.062. Because the population means are equal, the interval spans across 0.

Let us also briefly consider the χ^2 test. Table 5.20 shows a contingency table for our dataset. The table contains no cells where theoretical frequencies are less than 5 and hence the usage of the χ^2 test is justified. The test gives the value of χ^2 of 7.513. For $g = 4$ the risk of rejecting the zero hypothesis that there is no difference in the frequency of scores between Location 1 and Location 2 for question type C is 0.111. Since this is greater than 0.05, we accept the zero hypothesis and reject the alternative one. Finally, the value of the Cramer's V is 0.269.

Location/Score	1	2	3	4	5	Total
1	7	4	5	13	23	52
2	8	7	11	15	11	52
Total	15	11	16	28	34	104

Table 5.20: The scores for question type C at Location 1 against Location 2 for a sample of 104 evaluations

By comparing the population means and the frequencies of scores we thus cannot confirm that the scores differ between the configurations where the evaluators were explicitly forced to use an intrinsic reference frame fixed by the robot or where they could choose between a relative reference frame fixed by the robot and an intrinsic reference frame fixed by another reference object. This confirms that the potential changes of the reference frame by evaluators did not have a statistically significant effect on the evaluation.

5.3.5.6 Discretised and nominal concepts

In Section 5.3.4, page 231 we discussed the performance of *pDialogue* using classifiers whose target classes were nominal intervals created automatically from continuous

numeric attributes (*LO_x*, *LO_y*, *REFO_x* and *REFO_y*). These classifiers were used in question types C and D. We compared the performance of the system on these question types with its performance on question type A where it was using a classifier whose target class was genuinely nominal (*Relation*). We wanted to show that both the human evaluated performance of the system and the classifier accuracy were balanced in both scenarios. This would confirm that numeric target classes were discretised to an optimal number of nominal intervals. We claimed that the evaluated performance values for question types C and D are comparable but slightly higher than the performance values for question type A. We can use the t-test to support this claim statistically. To minimise the influence of other factors, we only consider the evaluations from Location 2.

Because we can only compare two sets of scores at once we have to perform the test twice. First we compare the evaluation scores for question type A with the scores for question type C. The size of both samples is $n_1 = n_2 = 52$. Their means and variances are $\bar{x}_1 = 2.81$, $s_1^2 = 1.963$, $\bar{x}_2 = 3.27$, $s_2^2 = 1.849$. The Levene's test of equality of population variances gives the value of $F = 0.044$. The risk of rejecting the zero hypothesis for such value of F for $g_1 = 1$ and $g_2 = 102$ degrees of freedom is $\alpha = P = 0.833 > 0.05$. The zero hypothesis cannot be rejected and a t-statistic can be calculated. This is $t = -1.705$ and its corresponding $\alpha = 2P$ value for $g = 102$ degrees of freedom is $0.91 > 0.05$. It follows that the zero hypothesis of the t-test cannot be rejected. The difference between the population means M_{x1} and M_{x2} for the evaluation scores for question types A and C are not statistically significant. With 95% likelihood we find the difference in the range between -0.998 and 0.075.

Next we compare the evaluation scores for question type A with the scores for question type D. Both samples are of size $n_1 = n_2 = 52$. Their means and variances are $\bar{x}_1 = 2.81$, $s_1^2 = 1.963$ and $\bar{x}_2 = 4.15$, $s_2^2 = 1.270$. The value of the F statistic obtained in the Levene's test is 5.506. Its corresponding value of $\alpha = P$ for $g_1 = 1$ and

$g_2 = 102$ degrees of freedom is $0.021 \leq 0.05$ which means that the zero hypothesis can be rejected. The population variances σ_1^2 and σ_2^2 are thus not equal and the standard t-test cannot be performed. However, we can still use an approximation of the t-test known as the Welch-Satterthwaite method (Sagadin, 2003, Section 12.4.6.2, page 258ff.). According to this method we calculate the t-statistic as before but we label it t' which is in this case -5.399. t' is not compared to the t-distribution for $g = n_1 + n_2 - 2 = 102$ degrees of freedom but to a t-distribution with an adjusted degrees of freedom $g' = 97.526 = 98$ (Sagadin, 2003, Equation 259, page 258). For $t' = -5.399$ the value of $\alpha = 2P$ for $g = 98$ degrees of freedom is 0.000 and hence the zero hypothesis can be rejected. It follows that the population means M_{x1} and M_{x2} of scores for question types A and D are not equal. The population mean M_{x2} is greater than the population mean M_{x1}. With a probability of 95% we find the difference of population means in the interval of 0.852 to 1.841. This means that we can expect the means to differ almost by 1 to 2 points.

It follows that the system performs better on question type D than on question types A and C where the performance is not statistically significantly different. Returning to the question of creating nominal intervals from continuous numeric variables, we can confirm that the method was well chosen.

5.3.6 Summary

In this part of the chapter we discussed the human evaluation of *pDialogue*, a system that uses the knowledge from the classifiers to perform motion commands and to answer questions about the location of objects. Due to the technical limitations the performance of the system on the motion commands could not be tested. The setting of answering questions is linguistically different from the setting of generating descriptions. We have shown that humans evaluated the performance of the system comparable to the performance of *pDescriber* which generates descriptions and which

most closely resembles the setting in which the linguistic and non-linguistic data was collected.

We started our discussion by describing the evaluation experiment which consisted of a set of pre-defined questions, the answers to which were evaluated by 13 evaluators. The questions and the locations at which they were asked were carefully selected to concentrate on various properties interesting for the investigation. We proceeded by measuring the agreement between the evaluators in terms of the correlation of their scores and concluded that the majority of evaluators show medium to high agreement with the rest of the group. There was a group of three evaluators whose agreement was quite low. A similar trend was also observed in the evaluation of *pDescriber*. We concluded that the level of agreement was good, considering that the evaluation task is difficult and considering that the system does not always generate the same response.

The performance of the system was evaluated at two levels. At the first level we examined evaluator scores to demonstrate how the performance of the system generating these descriptions compares to human performance. The varying configurations in which the robot was placed ensured that all factors influencing the performance of the system were covered in this test. The evaluated performance of the system is between 43.51% and 56.92% and is lower than the accuracy of underlying classifiers. It is also slightly lower than the performance of *pDescriber* (59.28%). It is important to bear in mind that each of these measures is slightly different and hence a direct comparison is not possible. Discussing the performance on question type B we pointed out that the system generated the best referring expression of relation between two objects in 28.88% of cases by only relying on low level representation of the scene. This is an encouraging result which should be supported by further experiments. Finally, we have shown that the evaluated performance of the system using classifiers whose target concepts were automatically converted from continuous numeric to nominal attributes was comparable to the performance of the system using classifiers with orig-

inally nominal target concepts. We concluded that the method of discretisation was appropriate in terms of the number of the nominal categories created.

At the second level we evaluated the performance of the system using statistical tests, the t-test and the χ^2 test, to confirm that the system performs differently across different configurations. We constructed these configurations so that we could directly evaluate the performance of the system on issues that evaluators reported problematic during the evaluation of *pDescriber* (Section 5.2.5). For example, some evaluators disliked descriptions of objects which were not in the "vision field" of the speaker or the robot. The statistical tests showed that the difference in evaluator scores between the two settings is statistically significant. Equally statistically significant proved to be the difference in scores of descriptions containing projective and topological relations. We concluded that topological relations should be modelled differently. Another test was performed to show whether the non-intended shift of perspective from the robot or the speaker to the reference object had an effect on the evaluation scores. We concluded that the scores from the configuration where the reference frame was fixed and the configuration where the evaluators could change it to the reference object while interpreting a description did not differ enough to be statistically significant. Finally, we examined the population means of scores for different question types. The t-test has shown that the sample scores for question types A and C are not statistically significantly different but those for question types A and D are.

5.4 Conclusion

This completes the evaluation of *pDescriber* and *pDialogue* by humans. We have shown that the performance of the systems is encouraging. The highest performance is achieved on the motion categories which were evaluated for *pDescriber* only. It was estimated to be 93.47% (the mean over all motion categories for all evaluators in Table 5.8). The performance on these categories turns out higher than the accuracy of the underly-

ing classifiers obtained through a 10-fold cross-validation. The system performance on the *Relation* category is 59.28% for *pDescriber* (Table 5.8) and 53.28% for *pDialogue* (the mean over all question types in Table 5.14). These figures translate to an acceptable performance but without doubt with some room for improvement. We discussed and evaluated the performance issues both qualitatively (*pDescriber*) and quantitatively (*pDialogue*). We concluded that when generating descriptions of relations the system over-generates in certain situations: this is either because the system does not incorporate certain properties of the environment or because of the data sparseness, for example in the case of topological relations.

The discussion has also shown that the way the classifier and non-classifier knowledge was integrated in *pDialogue* and *pDescriber* had a positive effect on the performance of the system. Finally, we have shown that the system can perform well in a different environment from the one in which the datasets for machine learning were collected and with a different set of humans interacting with it. This means that it is not tuned to a particular environment or people and is not suffering from systematic errors arising from it.

Chapter 6

Conclusion

The aim of this work was to show how the semantics of spatial expressions can be learned from low level data that a mobile robot has about itself and about its environment. We argued that the learning can be considered successful if the robot is able to replicate the descriptions that it has learned offline in new scenes in such a way that they appear natural to a human observer. This criteria resembles a simplified Turing test (Russell and Norvig, 2003, pages 2–3). For this reason, evaluation, both of the machine learning and the performance of the final systems, is central to our undertaking. It is also through evaluation that our work is most distinct from other related approaches where the systems have only been evaluated qualitatively (Section 2.3). Furthermore, the evaluation results tell us more than just whether our undertaking is successful. We discuss the performance of the robot by varying certain conditions. The differences in performance between such configurations of conditions represent valuable findings for the future undertakings of interacting robotic and natural language systems.

In this final chapter we summarise the observations and conclusions that we made and discuss possibilities for future work.

We started our exploration by outlining the ideas and tools that precede our work. We gave a summary of lexical semantics of descriptions that we expect our system to learn. These can be grouped into two categories which we call descriptions of motion

and descriptions of object relations. Their common property is that their semantics are partly indexical which means their full meaning can only be constructed by grounding them in the physical environment in which they are used. Because their interpretation relies on human perception of space we call them spatial descriptions. Excluding their indexical nature, descriptions of motion and topological descriptions of spatial relations are not semantically ambiguous. However, this is not true for projective descriptions of spatial relations which consistently have two interpretations depending on the orientation of the reference frame which is an integral part of their semantic structure.

Rather than using an existing model of semantics of spatial relations we proposed to learn the model from a corpus of natural language descriptions and representation of space available to a mobile robot. There are two reasons for this. The theoretical models of spatial descriptions proposed in the literature may not be immediately applicable on the data structures that are available to a mobile robot. Secondly, the decades of research in robotics have shown that pre-defined models of robotic knowledge by humans fail to perform well in a robot that is truly autonomous. This can only be achieved if the robot is able to learn from the environment so that it can adapt to its ever changing conditions. For example, the robotic system that we use in our experiments implements a technique known as SLAM through which the robot builds a global map of its environment based on its observations. An interesting research question is whether this approach can be extended to learning of lexical semantics of words. Can general semantic properties of spatial words and the values of their indexical parameters be also estimated or learned from the robot's linguistic and non-linguistic observations?

The robot's perception is different from human perception. The former relies on quantitative representations of space that were introduced by physical sciences. On the other hand, human perception and cognition which are reflected in the semantics

of spatial words operate on concepts such as colours, regions, typical patterns of behaviour, most of which are defined associatively. The wealth of information available to a component such as SLAM is quite poor in comparison to the wealth of information available to a human. For example, the SLAM component exclusively relies on observations from laser scans which come as a set of distances between the robot and some random points in space. These observations are enough to build a coherent map that can be used for navigation. Such information may be poor, but it is extremely precise. The challenge of this project is to show whether highly abstract human concepts can be induced from the poor stimulus available to the robot.

The learning of linguistic concepts is guided or supervised. This means that words are assigned to categories before they are applied to a learner. The learner therefore starts with more than just a string of unidentified words or even an acoustic waveform representing an utterance. Equally, for each learning setting we only include those properties of the environment and the robot that we consider important for the semantics of words that we want to learn. These properties are taken as they are available on the robot and were not adjusted to a particular model of spatial expressions. All values are normalised according to the current maximum values which allows their application in different configurations and numeric attributes are discretised to nominal classes if they are used as target concepts as required by our learners. Linguistic and non-linguistic observations are matched automatically and represented as instances which are then supplied to the learners.

We evaluated the performance of the classifiers that were learned using measures such as accuracy, the κ coefficient and ROC and Precision-Recall graphs. The evaluation was directed toward showing the differences in their performance in respect to the choices that were brought to the learning experiment. We worked with different learners (Naive-Bayes and J48), datasets (*Simple* and *All*), instance selection and creation mechanisms (*Time-shifted*, *Zero-Time-shifted*, *Not-Time-shifted* datasets) and num-

ber of discretised classes of numeric attributes. For an overall summary of results see Section 3.6, page 134.

In order to test the performance of classifiers on human evaluators they had to be integrated in simplified language generation and question answering systems called *pDescriber* and *pDialogue*. The performance of these systems was evaluated at different levels. We estimated agreement between the evaluators to measure if they can be considered as a single body and whether the systems have captured the desired generalisations. The accuracy of the systems was compared with the accuracies of the underlying classifiers. We discussed some qualitative observations of cases where the systems underperformed. Finally, we tested the results from a few configurations with statistical tests to confirm or reject their difference. The summaries of the results for both systems are given in Section 5.2.6 and 5.3.6 respectively. Overall we concluded that the systems have achieved an encouraging level of performance which means that semantics of spatial words can be learned from low-level robotic data (Section 5.4).

pDescriber and *pDialogue* perform very well on generating conceptually simpler descriptions of motion but slightly less well on descriptions of object relations where improvements could be introduced in the future. These improvements can be of two types. The first type includes improvements related to our model of spatial expressions. The classification of object relations would improve if objects were represented as richer structures rather than just points in a two-dimensional space. We mentioned that object shape, height and orientation may also be important for selecting the best spatial relation (Section 5.2.5.2). These structures could be learned separately from laser scans by discovering patterns in the arrangements of points.

Learning whether a particular projective relation should be interpreted according to a relative reference frame or an intrinsic reference frame is a very difficult task because the reference frame is not distinguished in the properties of the environment nor linguistically. When a hearer is unsure which reference frame is intended, they

resolve this uncertainty by grounding the reference object in the scene and choosing the interpretation of the relation that best satisfies its semantic and pragmatic constraints. Alternatively, they may ask the speaker a clarification question. Therefore, the intended reference frame can be established not from a description but from the discourse. Perhaps another set of experiments could be constructed where the describers would be asked to use linguistic cues such as "from my/your/robot's point of view" which would allow us to represent the intended reference frame as a feature in the learning process.

We concluded that overall evaluators did not like descriptions of objects that were not in the "vision field" of the robot (Section 5.3.5.3). The system did not over-generate because of our model of space but because such descriptions were present in the dataset from which the classifiers were built. Therefore, this error is not due to the shortcomings of our machine learning model but due to the differences in human intuition about the descriptions.

The improvements of the second type include linking the semantic concepts that were learned by our system with new modalities. A discussion of the design of *pDescriber* and *pDialogue* reveals that a correct classification of a relation is not the only condition for a successfully generated description. Selection of the right objects is also important. In addition to their spatial properties discussed above, objects would also have to be represented for properties such as colour, type and typical actions that they allow (Section 4.4.4, page 154). Ideally, this information should be learned rather than be pre-specified. Combining information from different sources would require a more complex knowledge representation. Finally, sometimes reference objects are selected because they have been discussed in the discourse before, they have been grounded and therefore they are good candidates to describe the location of another object. This means that generation of spatial descriptions should be integrated with a dialogue system. Our systems do not attempt to be embodied conversational agents which are able

to reason about such rich knowledge. Therefore, they fall sometimes short regardless how well or bad the underlying classifiers perform. The evaluators consider entire utterances rather than just categories of words.

Overall, the current systems provide important insight into the issues related to interfacing information from different modalities. For example, we use the information from localisation and map building to learn the semantics of spatial relations. The flow of information could also be reversed. Perhaps the information from linguistic descriptions could be used used by the localisation component if the map of the environment is incomplete. The project connects robotic and natural language communities both of which investigate the same topic, inducing representations of space, but approach it from different perspectives and research goals in mind. Valuable lessons have been learned but no doubt much still lies ahead for the future.

Appendix A: Classifier performance per class

Dataset	Class	TP Rate		FP Rate		Precision		Recall		F-Measure		ROC Area	
		J48	NB	J48	NB	J48	NB	J48	NB	J48	NB	J48	NB
Verb													
Simple-Not-Time-shifted	moving	0.957	0.674	0.194	0.083	0.863	0.912	0.957	0.674	0.907	0.775	0.845	0.932
	stopped	0.806	0.917	0.043	0.326	0.935	0.688	0.806	0.917	0.866	0.786	0.845	0.936
All-Time-shifted	going	0.281	0.103	0.06	0.029	0.406	0.345	0.281	0.103	0.332	0.158	0.715	0.71
	edging	0	0	0	0	0	0	0	0	0	0	0.523	0.572
	continuing	0	0	0	0	0	0	0	0	0	0	0.399	0.053
	reversing	0.221	0.039	0.018	0.007	0.405	0.231	0.221	0.039	0.286	0.067	0.813	0.823
	creeping	0	0	0.004	0	0	0	0	0	0	0	0.685	0.736
	turning	0.55	0.382	0.163	0.08	0.552	0.635	0.55	0.382	0.551	0.477	0.756	0.761
	moving	0.604	0.778	0.272	0.459	0.579	0.512	0.604	0.778	0.591	0.617	0.674	0.686
	stopped	0.981	0.972	0.08	0.079	0.682	0.682	0.981	0.972	0.805	0.802	0.951	0.96
Direction													
Simple-Not-Time-shifted	backward	0.857	0.143	0	0	1	1	0.857	0.143	0.923	0.25	0.929	0.952
	forward	1	0.652	0.153	0.068	0.719	0.789	1	0.652	0.836	0.714	0.9	0.878
	none	0.827	0.923	0.033	0.467	0.977	0.774	0.827	0.923	0.896	0.842	0.894	0.817
All-Time-shifted	stopped	0	1	0	0.195	0	0.007	0	1	0	0.014	0.802	0.826
	spot	0.25	0.083	0.023	0.004	0.367	0.5	0.25	0.083	0.298	0.143	0.76	0.817
	backward	0.7	0.411	0.052	0.029	0.694	0.702	0.7	0.411	0.697	0.518	0.881	0.864
	forward	0.659	0.567	0.189	0.204	0.648	0.595	0.659	0.567	0.653	0.581	0.804	0.784
	none	0.664	0.338	0.305	0.421	0.65	0.407	0.664	0.338	0.657	0.369	0.711	0.425
Heading													
Simple-Not-Time-shifted	right	0.9	1	0	0.014	1	0.909	0.9	1	0.947	0.952	0.95	0.999
	left	1	1	0.015	0	0.938	1	1	1	0.968	1	0.993	1
	none	0.982	0.982	0.04	0	0.982	1	0.982	0.982	0.982	0.991	0.971	0.996
All-Time-shifted	anticlockwise	0	0	0.001	0	0	0	0	0	0	0	0.799	0.797
	clockwise	0.238	0	0.003	0	0.556	0	0.238	0	0.333	0	0.726	0.809
	straight_ahead	0	0	0	0	0	0	0	0	0	0	0.449	0.242
	hard	0	1	0	0.196	0	0.004	0	1	0	0.007	0.291	0.871
	straight_line	0	0	0.001	0.001	0	0	0	0	0	0	0.414	0.388
	around	0	0	0	0.001	0	0	0	0	0	0	0.498	0.424
	straight	0.013	0	0.01	0	0.071	0	0.013	0	0.021	0	0.575	0.543
	180	0	0	0	0.001	0	0	0	0	0	0	0.44	0.573

Continued on next page

Continued from previous page

Dataset	Class	TP Rate		FP Rate		Precision		Recall		F-Measure		ROC Area	
		J48	NB	J48	NB	J48	NB	J48	NB	J48	NB	J48	NB
	right	0.535	0.309	0.093	0.05	0.552	0.572	0.535	0.309	0.544	0.401	0.751	0.76
	left	0.674	0.402	0.109	0.052	0.619	0.672	0.674	0.402	0.645	0.503	0.793	0.784
	none	0.812	0.62	0.344	0.541	0.714	0.548	0.812	0.62	0.759	0.582	0.752	0.589
Manner													
Simple-Not-Time-shifted	fast	0.2	0.2	0.052	0	0.2	1	0.2	0.2	0.2	0.333	0.7	0.584
	moderately	0.2	0.4	0.052	0.104	0.2	0.2	0.2	0.4	0.2	0.267	0.74	0.712
	slowly	0.364	0	0.042	0.014	0.571	0	0.364	0	0.444	0	0.71	0.729
	none	0.852	0.918	0.619	0.667	0.8	0.8	0.852	0.918	0.825	0.855	0.69	0.604
All-Time-shifted	quickly	0.042	0	0.007	0	0.091	0	0.042	0	0.057	0	0.668	0.688
	walking_pace	0	0	0	0.017	0	0	0	0	0	0	0.447	0.034
	imperceptible	0	0	0.001	0.019	0	0	0	0	0	0	0.496	0.903
	tightly	0	0	0.004	0	0	0	0	0	0	0	0.494	0.755
	gently	0.038	0	0.015	0	0.13	0	0.038	0	0.059	0	0.546	0.579
	rapidly	0	0	0	0	0	0	0	0	0	0	0.427	0.714
	fast	0.224	0.342	0.022	0.031	0.362	0.377	0.224	0.342	0.276	0.359	0.648	0.78
	moderately	0.11	0.011	0.031	0.011	0.192	0.063	0.11	0.011	0.14	0.019	0.563	0.695
	slowly	0.38	0	0.142	0	0.436	0	0.38	0	0.406	0	0.701	0.605
	none	0.797	0.914	0.591	0.892	0.643	0.578	0.797	0.914	0.712	0.708	0.658	0.588
Relation													
Simple	behind	0.719	0.754	0.05	0.032	0.788	0.86	0.719	0.754	0.752	0.804	0.873	0.931
	in_front_of	0.638	0.741	0.109	0.073	0.607	0.729	0.638	0.741	0.622	0.735	0.813	0.928
	to_the_right_of	0.839	0.816	0.084	0.058	0.82	0.866	0.839	0.816	0.83	0.84	0.879	0.907
	to_the_left_of	0.789	0.882	0.079	0.099	0.789	0.77	0.789	0.882	0.789	0.822	0.907	0.941
All	next	0	0	0.002	0	0	0	0	0	0	0	0.49	0.821
	after	0	0	0	0	0	0	0	0	0	0	0.436	0.436
	near	0.133	0.067	0.02	0.015	0.143	0.1	0.133	0.067	0.138	0.08	0.668	0.816
	parallel	0	0	0.003	0.002	0	0	0	0	0	0	0.361	0.358
	opposite	0	0	0	0	0	0	0	0	0	0	0.484	0.611
	facing	0.333	0.333	0.01	0.023	0.4	0.222	0.333	0.333	0.364	0.267	0.77	0.84
	front	0.683	0.577	0.096	0.056	0.636	0.717	0.683	0.577	0.659	0.64	0.83	0.903
	far	0.286	0.714	0.015	0.051	0.308	0.244	0.286	0.714	0.296	0.364	0.659	0.899
	right	0.76	0.727	0.083	0.1	0.75	0.704	0.76	0.727	0.755	0.716	0.857	0.864
	left	0.804	0.816	0.08	0.113	0.78	0.719	0.804	0.816	0.792	0.764	0.902	0.908
	behind	0.711	0.772	0.043	0.033	0.786	0.838	0.711	0.772	0.747	0.804	0.869	0.907

Continued on next page

Continued from previous page

Dataset	Class	TP Rate		FP Rate		Precision		Recall		F-Measure		ROC Area	
		J48	NB	J48	NB	J48	NB	J48	NB	J48	NB	J48	NB
	close	0.391	0.043	0.028	0.01	0.346	0.143	0.391	0.043	0.367	0.067	0.792	0.804
Delta-Heading, 7 bins													
Simple-Not-Time-shifted	-0.6001...-0.4286	0	0	0.025	0	0	0	0	0	0	0	0.941	0.9
	-0.4286...-0.2572	0.5	0.5	0.038	0.038	0.4	0.4	0.5	0.5	0.444	0.444	0.962	0.949
	-0.2572...-0.0857	0.6	0.6	0	0.026	1	0.6	0.6	0.6	0.75	0.6	0.66	0.771
	-0.0857...0.0857	0.949	0.949	0.043	0.043	0.982	0.982	0.949	0.949	0.966	0.966	0.941	0.928
	0.0857...0.2572	0.75	0.625	0.095	0.068	0.462	0.5	0.75	0.625	0.571	0.556	0.941	0.966
	0.2572...0.4286	0	0	0.026	0.064	0	0	0	0	0	0	0.934	0.939
	0.4286...0.6001	0	0	0	0	0	0	0	0	0	0	–	–
All-Time-shifted	-0.8791...-0.6279	0	0	0	0	0	0	0	0	0	0	0.197	0.367
	-0.6279...-0.3768	0.067	0	0.001	0.001	0.5	0	0.067	0	0.118	0	0.437	0.786
	-0.3768...-0.1256	0.047	0.503	0.007	0.088	0.438	0.397	0.047	0.503	0.085	0.444	0.674	0.814
	-0.1256...0.1256	0.967	0.812	0.912	0.419	0.751	0.847	0.967	0.812	0.846	0.829	0.61	0.797
	0.1256...0.3768	0.101	0.529	0.025	0.103	0.38	0.437	0.101	0.529	0.159	0.478	0.624	0.815
	0.3768...0.6279	0	0	0	0	0	0	0	0	0	0	0.621	0.81
	0.6279...0.8791	0	0	0	0	0	0	0	0	0	0	0.273	0.695
Speed, 7 bins													
Simple-Not-Time-shifted	-1.0055...-0.7182	0	0	0	0	0	0	0	0	0	0	0.463	0.79
	-0.7182...-0.4309	0	0	0	0	0	0	0	0	0	0	0.463	0.778
	-0.4309...-0.1436	1	1	0.038	0.026	0.571	0.667	1	1	0.727	0.8	0.963	0.994
	-0.1436...0.1436	0.933	0.933	0.216	0.108	0.84	0.913	0.933	0.933	0.884	0.923	0.88	0.908
	0.1436...0.4309	0.714	0.857	0.115	0.098	0.682	0.75	0.714	0.857	0.698	0.8	0.76	0.921
	0.4309...0.7182	0	0.143	0.027	0.053	0	0.2	0	0.143	0	0.167	0.447	0.723
	0.7182...1.0055	0	0	0.013	0.013	0	0	0	0	0	0	0.559	0.629
All-Time-shifted	-1.0742...-0.7673	0	0.176	0.006	0.009	0	0.316	0	0.176	0	0.226	0.773	0.895
	-0.7673...-0.4604	0.37	0.296	0.023	0.016	0.385	0.421	0.37	0.296	0.377	0.348	0.909	0.928
	-0.4604...-0.1535	0.742	0.735	0.091	0.075	0.452	0.495	0.742	0.735	0.562	0.591	0.892	0.915
	-0.1535...0.1535	0.536	0.588	0.072	0.141	0.828	0.728	0.536	0.588	0.651	0.65	0.776	0.78
	0.1535...0.4604	0.85	0.756	0.319	0.275	0.534	0.541	0.85	0.756	0.656	0.631	0.795	0.809
	0.4604...0.7673	0.277	0.266	0.028	0.037	0.583	0.5	0.277	0.266	0.375	0.347	0.791	0.815
	0.7673...1.0742	0.229	0.313	0.011	0.014	0.423	0.429	0.229	0.313	0.297	0.361	0.756	0.818
LO_x, 7 bins													
Simple	-1.0...-0.7143	0	0	0	0.59	0	0	0	0	0	0	–	–
	-0.7143...-0.4286	0.353	0.059	0.011	0	0.667	1	0.353	0.059	0.462	0.111	0.889	0.821

Continued on next page

Continued from previous page

Dataset	Class	TP Rate		FP Rate		Precision		Recall		F-Measure		ROC Area	
		J48	NB	J48	NB	J48	NB	J48	NB	J48	NB	J48	NB
	-0.4286…-0.1429	0.523	0.169	0.117	0.085	0.576	0.379	0.523	0.169	0.548	0.234	0.722	0.714
	-0.1429…0.1429	0.784	0.307	0.237	0.068	0.605	0.675	0.784	0.307	0.683	0.422	0.748	0.797
	0.1429…0.4286	0.728	0.21	0.132	0.076	0.694	0.531	0.728	0.21	0.711	0.301	0.799	0.768
	0.4286…0.7143	0.136	0.045	0.027	0.039	0.3	0.091	0.136	0.045	0.187	0.061	0.733	0.739
	0.7143…1.0	0	0	0.004	0.004	0	0	0	0	0	0	0.881	0.631
All	-1.0…-0.7143	0.286	0	0.002	0.003	0.667	0	0.286	0	0.4	0	0.821	0.688
	-0.7143…-0.4286	0.204	0.551	0.009	0.123	0.667	0.276	0.204	0.551	0.313	0.367	0.822	0.825
	-0.4286…-0.1429	0.553	0.14	0.145	0.086	0.546	0.339	0.553	0.14	0.55	0.198	0.754	0.723
	-0.1429…0.1429	0.797	0.779	0.283	0.328	0.608	0.567	0.797	0.779	0.69	0.657	0.788	0.777
	0.1429…0.4286	0.652	0.6	0.111	0.121	0.66	0.62	0.652	0.6	0.656	0.61	0.799	0.797
	0.4286…0.7143	0.086	0	0.012	0.012	0.3	0	0.086	0	0.133	0	0.804	0.783
	0.7143…1.0	0	0	0.002	0.002	0	0	0	0	0	0	0.823	0.739
LO_y, 7 bins													
Simple	-1.0…-0.7143	0	0	0	0.59	0	0	0	0	0	0	–	–
	-0.7143…-0.4286	0	0	0	0	0	0	0	0	0	0	–	–
	-0.4286…-0.1429	0.656	0.131	0.051	0.069	0.784	0.348	0.656	0.131	0.714	0.19	0.864	0.653
	-0.1429…0.1429	0.857	0.371	0.35	0.184	0.806	0.774	0.857	0.371	0.831	0.502	0.754	0.709
	0.1429…0.4286	0.643	0.095	0.059	0.013	0.659	0.571	0.643	0.095	0.651	0.163	0.78	0.796
	0.4286…0.7143	0	0	0	0	0	0	0	0	0	0	–	–
	0.7143…1.0	0	0	0	0	0	0	0	0	0	0	–	–
All	-1.0…-0.7143	0	0	0	0.064	0	0	0	0	0	0	–	–
	-0.7143…-0.4286	0	0	0	0	0	0	0	0	0	0	–	–
	-0.4286…-0.1429	0.77	0.204	0.045	0.035	0.791	0.561	0.77	0.204	0.78	0.299	0.889	0.804
	-0.1429-0.1429	0.882	0.451	0.286	0.166	0.853	0.836	0.882	0.451	0.867	0.586	0.801	0.733
	0.1429…0.4286	0.641	0.039	0.052	0.01	0.71	0.444	0.641	0.039	0.673	0.071	0.831	0.711
	0.4286…0.7143	0	0	0	0.505	0	0	0	0	0	0	0.442	0.47
	0.7143…1.0	0	0	0	0	0	0	0	0	0	0	–	–
REFO_x, 7 bins													
Simple	-1.0…-0.7143	0	0	0	0.072	0	0	0	0	0	0	–	–
	-0.7143…-0.4286	0.5	0	0.007	0.004	0.5	0	0.5	0	0.5	0	0.912	0.858
	-0.4286…-0.1429	0.1	0	0.004	0.024	0.75	0	0.1	0	0.176	0	0.66	0.579
	-0.1429…0.1429	0.974	0.922	0.847	0.788	0.723	0.727	0.974	0.922	0.83	0.813	0.632	0.689
	0.1429…0.4286	0.079	0	0.013	0.004	0.5	0	0.079	0	0.136	0	0.614	0.54
	0.4286…0.7143	0.273	0	0.004	0	0.75	0	0.273	0	0.4	0	0.668	0.774

Continued on next page

Continued from previous page

Dataset	Class	TP Rate		FP Rate		Precision		Recall		F-Measure		ROC Area	
		J48	NB	J48	NB	J48	NB	J48	NB	J48	NB	J48	NB
	0.7143…1.0	0	1	0	0.011	0	0.4	0	1	0	0.571	0.923	1
All	-1.0…-0.7143	0	1	0	0.148	0	0.021	0	1	0	0.042	0.984	0.984
	-0.7143…-0.4286	0.52	0	0.015	0.003	0.591	0	0.52	0	0.553	0	0.858	0.775
	-0.4286…-0.1429	0.135	0	0.034	0.005	0.345	0	0.135	0	0.194	0	0.702	0.373
	-0.1429…0.1429	0.954	0.897	0.731	0.667	0.755	0.761	0.954	0.897	0.843	0.823	0.663	0.656
	0.1429…0.4286	0.044	0.015	0.016	0.004	0.25	0.333	0.044	0.015	0.075	0.028	0.537	0.51
	0.4286…0.7143	0.133	0	0.007	0	0.333	0	0.133	0	0.19	0	0.786	0.688
	0.7143…1.0	0	1	0.002	0.005	0	0.4	0	1	0	0.571	0.945	0.998
REFO_y, 7 bins													
Simple	-1.0…-0.7143	0	0	0	0.072	0	0	0	0	0	0	–	–
	-0.7143…-0.4286	0	0	0	0	0	0	0	0	0	0	–	–
	-0.4286…-0.1429	0.542	0	0.024	0	0.684	0	0.542	0	0.605	0	0.731	0.544
	-0.1429…0.1429	0.966	0.954	0.55	0.775	0.913	0.88	0.966	0.954	0.939	0.915	0.724	0.645
	0.1429…0.4286	0.063	0	0.023	0	0.143	0	0.063	0	0.087	0	0.599	0.532
	0.4286…0.7143	0	0	0	0	0	0	0	0	0	0	–	–
	0.7143…1.0	0	0	0	0	0	0	0	0	0	0	–	–
All	-1.0…-0.7143	0	0	0	0	0	0	0	0	0	0	–	–
	-0.7143…-0.4286	0	0	0	0	0	0	0	0	0	0	0.465	0.078
	-0.4286…-0.1429	0.5	0.021	0.019	0.003	0.686	0.333	0.5	0.021	0.578	0.039	0.815	0.579
	-0.1429…0.1429	0.95	0.892	0.538	0.585	0.896	0.882	0.95	0.892	0.922	0.887	0.739	0.719
	0.1429…0.4286	0.296	0	0.042	0.005	0.4	0	0.296	0	0.34	0	0.684	0.45
	0.4286…0.7143	0	1	0	0.148	0	0.021	0	1	0	0.042	0.465	0.875
	0.7143…1.0	0	0	0	0	0	0	0	0	0	0	–	–

References

Artstein, Ron, and Massimo Poesio. 2005. Kappa^3 = alpha (or beta). Technical report, University of Essex Department of Computer Science. Available at: http://cswww.essex.ac.uk/technical-reports/2005/csm-437.pdf.

Baldridge, Jason, and Geert-Jan M. Krujiff. 2002. Coupling CCG and hybrid logic dependency semantics. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, 319–326. Philadelphia, PA, USA.

Bartko, John J., and William T. Jr. Carpenter. 1976. On the methods and theory of reliability. *Journal of Nervous and Mental Disease* 163:307–317.

Bayes, Thomas. 1763. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. *Philosophical Transactions, Giving Some Account of the Present Undertakings, Studies and Labours of the Ingenious in Many Considerable Parts of the World* 370–418.

Ben-David, Arie. 2007. A lot of randomness is hiding in accuracy. *Engineering Applications of Artificial Intelligence* 20:875–885.

Bos, Johan, Ewan Klein, and Tetsushi Oka. 2003. Meaningful conversation with a mobile robot. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 10)*, 71–74. Budapest.

Bosse, Michael, and Robert Zlot. 2008. Map matching and data association for large-scale two-dimensional laser scan-based SLAM. *The International Journal of Robotics Research* 27:667–691.

Bratko, Ivan. 2001. *Prolog programming for artificial intelligence*. Addison Wesley: Pearson Education, 3rd edition.

Carletta, Jean. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics* 2:249–254.

Cohen, Jakob. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20:37–64.

Coventry, Kenny. 2003. Spatial prepositions, spatial templates, and "semantic" versus "pragmatic" visual representations. In *Representing direction in language and space*, ed. Emile van der Zee and Jon Slack, Explorations in language and space, chapter 13, 255–267. Oxford University Press.

Coventry, Kenny R., Mercè Prat-Sala, and Lynn Richards. 2001. The interplay between geometry and function in the apprehension of Over, Under, Above and Below. *Journal of memory and language* 44:376–398.

Crangle, Colleen, and Patrick Suppes. 1994. *Language and learning for robots*. Stanford, California: CSLI Publications.

Curran, James R., Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proceedings of the ACL 2007 Demonstrations Sessions (ACL-07 demo)*, 29–32.

Dale, Robert, and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive science* 19:233–263.

van Deemter, Kees. 2006. Generating referring expressions that involve gradable properties. *Computational Linguistics* 32:195–222.

Dissanayake, M. W. M. G, P. M. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csorba. 2001. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotic and Automation* 17:229–241.

Fayyad, Usama M., and Keki B. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1022–1027. Chambery, France: Morgan Kaufmann: San Francisco.

Folkesson, John, Patric Jensfelt, and Henrik I. Christensen. 2005. Vision SLAM in the measurement subspace. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'05)*, 30–35.

Gapp, Klaus-Peter. 1994a. Basic meanings of spatial relations: computation and evaluation in 3D space. In *Proceedings of the twelfth national conference on Artificial Intelligence (AAAI'94)*, volume 2, 1393–1398. American Association for Artificial Intelligence, Menlo Park, CA, USA: AAAI Press/MIT Press.

Gapp, Klaus-Peter. 1994b. A computational model of the basic meanings of graded composite spatial relations in 3D space. In *Advanced geographic data modelling. Spatial data modelling and query languages for 2D and 3D applications (Proceedings of the AGDM'94)*, Publications on Geodesy 40, 66–79. Netherlands Geodetic Commission.

Gapp, Klaus-Peter. 1995. Angle, distance, shape, and their relationship to projective relations. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, 112–117. Mahwah, NJ: Lawrence Erlbaum.

Gorniak, Peter, and Deb Roy. 2004. Grounded semantic composition for visual scenes. *Journal of Artificial Intelligence Research* 21:429–470.

Harnad, Stevan. 1990. The symbol grounding problem. *Physica D* 42:335–346.

Herskovits, Annette. 1986. *Language and spatial cognition: an interdisciplinary study of the prepositions in English*. Cambridge: Cambridge University Press.

Horswill, Ian Douglas. 1998. Grounding mundane inference in perception. *Autonomous Robots* 5:63–77.

Kamp, Hans, and Uwe Reyle. 1993. *From discourse to logic: introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*. Studies in linguistics and philosophy. Dordrecht, London: Kluwer Academic.

Kelleher, John, and Fintan Costello. 2005. Cognitive representations of projective prepositions. In *Proceedings of the Second ACL-SIGSEM workshop on the linguistic dimensions of prepositions and their use in computational linguistics formalisms and applications*, 119–127. University of Essex, Colchester, United Kingdom: Association of Computational Linguistics.

Kelleher, John D., Geert-Jan M. Kruijff, and Fintan J. Costello. 2006. Proximity in context: an empirically grounded computational model of proximity for processing topological spatial expressions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 745–752. Sydney, Australia: Association for Computational Linguistics. Available at: http://www.aclweb.org/anthology/P06-1094.

Kohavi, Ron. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 1137–1143. Montreal, Canada: San Francisco: Morgan Kaufmann.

Kruijff, Geert-Jan, Hendrik Zender, Patric Jensfelt, and Henrik I. Christensen. 2006. Clarification dialogues in human-augmented mapping. In *Proceedings of the 1st ACM*

SIGCHI/SIGART conference on Human-robot interaction, 282–289. Salt Lake City, Utah, USA: ACM/IEEE International Conference on Human-Robot Interaction.

Krujiff, Geert-Jan M., Hendrik Zender, Patric Jensfelt, and Henrik I. Christensen. 2007. Situated dialogue and spatial organization: what, where...and why? *International Journal of Advanced Robotic Systems* 4:125–138. Special issue on human and robot interactive communication.

Kyriacou, Theocharis, Guido Bugmann, and Stanislao Lauria. 2005. Vision-based urban navigation procedures for verbally instructed robots. *Robotics and Autonomous Systems* 51:69–80.

Lauria, Stanislao, Guido Bugmann, Theocharis Kyriacou, Johan Bos, and Ewan Klein. 2001. Training personal robots using natural language instruction. *IEEE Intelligent Systems* 16:38–45.

Lauria, Stanislao, Guido Bugmann, Theocharis Kyriacou, and Ewan Klein. 2002a. Mobile robot programming using natural language. *Robotics and Autonomous Systems* 38:171–181.

Lauria, Stanislao, Theocharis Kyriacou, Guido Bugmann, Johan Bos, and Ewan Klein. 2002b. Converting natural language route instructions into robot-executable procedures. In *Proceedings of the 2002 IEEE International Workshop on Robot and Human Interactive Communication (Roman'02)*, 223–228. Berlin.

Leonard, John J., and Hugh F. Durrant-Whythe. 1991. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS'91*, 1442–1447. Osaka, Japan: IEEE/RSJ.

Levene, Howard. 1960. Robust tests for equality of variances. In *Contributions to probability and statistics*, ed. I. Olkin, 278–292. Stanford University Press, Palo Alto, CA.

Levinson, Stephen. 1996a. Frames of reference and Molyneux's question: crosslinguistic evidence. In *Language and space*, ed. Paul Bloom, Mary A. Peterson, Lynn Nadel, and Merrill F. Garrett, 109–169. Cambridge, MA: MIT Press.

Levinson, Stephen C. 1983. *Pragmatics*. Cambridge textbooks in linguistics. Cambridge: Cambridge University Press.

Levinson, Stephen C. 1996b. Relativity in spatial conception and description. In *Rethinking linguistic relativity*, ed. John J. Gumperz and Stephen C. Levinson, 13–45. Cambridge: Cambridge University Press.

Levinson, Stephen C. 2003. *Space in language and cognition: explorations in cognitive diversity*. Cambridge: Cambridge University Press.

Logan, Gordon D. 1994. Spatial attention and the apprehension of spatial relations. *Journal of Experimental Psychology: Human Perception and Performance* 20:1015–1036.

Logan, Gordon D. 1995. Linguistic and conceptual control of visual spatial attention. *Cognitive Psychology* 28:103–174.

Logan, Gordon D., and Daniel D. Sadler. 1996. A computational analysis of the apprehension of spatial relations. In *Language and space*, ed. Paul Bloom, Mary A. Peterson, Lynn Nadel, and Merrill F. Garrett, 493–530. Cambridge, MA: MIT Press.

Maillat, Didier. 2001. Directional PPs and reference frames in DRT. In *Proceedings of the ACL Workshop on temporal and spatial information processing*, volume 13, 1–8. Toulouse: Université de Toulouse – Le Mirail: Association for Computational Linguistics.

Maillat, Didier. 2003. The semantics and pragmatics of directionals: a case study in English and French. Doctoral Dissertation, Committee for Comparative Philology and General Linguistics, University of Oxford, Oxford, UK.

Manning, Christopher D., and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. The MIT Press.

Mavridis, Nikolaos, and Deb Roy. 2006. Grounded situation models for robots: where words and percepts meet. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006)*, 4690–4697. Beijing, China.

Miller, George A., and Philip N. Johnson-Laird. 1976. *Language and perception*. Cambridge: Cambridge University Press.

Mitchell, Tom M. 1997. *Machine learning*. McGraw-Hill Series in Computer Science. McGraw-Hill.

Mukerjee, Amitabha. 1998. Neat versus scruffy: a review of computational models of spatial expressions. In *Representation and processing of spatial expressions*, ed. Patrick Olivier and Klaus-Peter Gapp, 1–36. Mahwah, N.J.: Lawrence Erlbaum Associates.

Newman, Paul M. 2003. C4B mobile robotics. Lecture notes, University of Oxford. Available at: http://www.robots.ox.ac.uk/~pnewman/Teaching/C4CourseResources/C4BMobileRobots.pdf.

Newman, Paul M. 2006. MOOS – Mission Oriented Operating Suite. Technical report, Department of Ocean Engineering (MIT), Department of Engineering Science (Oxford University). Available at: http://www.robots.ox.ac.uk/~pnewman/papers/MOOS.pdf.

Newman, Paul M., and Hugh F. Durrant-Whyte. 2001. An efficient solution to the SLAM problem using geometric projections. In *Proceedings of the November 2001 SPIE conference*. Boston, USA.

Newman, Paul M., and John J. Leonard. 2003. Consistent, convergent, and constant-time SLAM. In *IJCAI*, ed. Georg Gottlob and Toby Walsh, 1143–1150. Morgan Kaufmann.

Nilsson, Nils J. (editor). 1984. Shakey the robot. Technical Report 323, SRI International: Artificial Intelligence Center, Computer Science and Technology Division.

Olivier, Patrick, and Jun-ichi Tsujii. 1994. A computational view of the cognitive semantics of spatial prepositions. In *Proceedings of the 32nd Annual Meeting of the Association of Computational Linguistics*, 303–309. New Mexico State University, Morgan Kaufmann Publishers.

Quinlan, J.R. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Quinlan, J.R. 1993. *C4.5: programs for machine learning*. San Francisco: Morgan Kaufmann.

Regier, Terry. 1996. *The human semantic potential: spatial language and constrained connectionism*. Cambridge, Massachusetts, London, England: MIT Press.

Regier, Terry, and Laura A. Carlson. 2001. Grounding spatial language in perception: an empirical and computational investigation. *Journal of Experimental Psychology: General* 130:273–298.

Rennie, Jason D. 2004. Derivation of the F-measure. Available at: http://people.csail.mit.edu/jrennie/writing.

van Rijsbergen, C.J. 1979. *Information retrieval*. London: Butterworths, 2nd edition.

Roy, Deb. 2005. Semiotic schemas: a framework for grounding language in action and perception. *Artificial Intelligence* 167:170–205.

Roy, Deb, Kai-Yuh Hsiao, and Nikolaos Mavridis. 2004. Mental imagery for a conversational robot. *IEEE Transactions on Systems, Man and Cybernetics–Part B: Cybernetics* 34:1374–1383.

Roy, Deb K. 2002. Learning visually-grounded words and syntax for a scene description task. *Computer speech and language* 16:353–385.

Russell, Stuart, and Peter Norvig. 2003. *Artificial intelligence: a modern approach*. Prentice Hall series in Artificial Intelligence. Upper Saddle River, New Jersey: Prentice Hall, 2nd edition.

Sagadin, Janez. 2003. *Statistične metode za pedagoge*. Obzorja, Maribor, Slovenia.

Smith, Randall C., and Peter Cheeseman. 1986. On the representation and estimation of spatial uncertainty. *The international journal of robotics research* 5:56–68.

Spärk Jones, Karen. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 28:11–21. Reprinted in Volume 60 Number 5 2004, 493-502.

Stachniss, Cyrill, Óscar Martínez-Mozos, Axel Rottmann, and Wolfram Burgard. 2005. Semantic labelling of places. In *Proceedings of the International Symposium of Robotics Research (ISRR)*. San Francisco, CA, USA.

Steels, Luc, and Jean-Christophe Baillie. 2003. Shared grounding of event descriptions by autonomous robots. *Robotics and Autonomous Systems* 43:163–173.

Stopp, Eva, Klaus-Peter Gapp, Gerd Herzog, Thomas Laengle, and Tim C. Lueth. 1994. Utilizing spatial relations for natural language access to an autonomous mobile robot. In *KI-94: Advances in artificial intelligence*, ed. B. Nebel and L. Dreschler-Fischer, volume 861/1994 of *Lecture Notes in Computer Science*, 39–50. Berlin, Heidelberg: Springer.

Sukkarieh, Jana Zuheir. 2000. Natural language for knowledge representation. Doctoral Dissertation, Computer Laboratory, Churchill College, University of Cambridge, Cambridge, UK.

Talmy, Leonard. 1983. How language structures space. In *Spatial orientation: theory, research, and application*, ed. Herbert L. Pick Jr. and Linda P. Acredolo, 225–282. New

York: Plenum Press. Based on the proceedings of a Conference on Spatial Orientation and Perception held on July 14-16, 1980, at the University of Minnesota, Minneapolis, Minnesota.

Talmy, Leonard. 2000. *Toward a cognitive semantics: concept structuring systems*, volume 1. Cambridge, Massachusetts: MIT Press.

Theobalt, Christian, Johan Bos, Tim Chapman, Arturo Espinosa-Romero, Mark Fraser, Gillian Hayes, Ewan Klein, Tetsushi Oka, and Richard Reeve. 2002. Talking to Godot: dialogue with a mobile robot. In *Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and System (IROS 2002)*, 1338–1343. Lausanne, Switzerland.

Vandeloise, Claude. 1986. *L'espace en français: sémantique des própositions spatiales*. Paris: Éditions de Seuil.

Vandeloise, Claude. 1991. *Spatial prepositions: a case study from French*. University of Chicago Press. English translation of L'espace en français by Anna R.K. Bosch.

Weng, Fuliang, Baoshi Yan, Zhe Feng, Florin Ratiu, Madhuri Raya, Brian Lathrop, Annie Lien, Rohit Mishra, Sebastian Varges, Feng Lin, Matthew Purver, Yao Meng, Harry Bratt, Tobias Scheideck, Zhaoxia Zhang, Badri Raghunathan, and Stanley Peters. 2007. CHAT to your destination. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, ed. Simon Keizer, Harry Bunt, and Tim Paek, 79–86. Antwerp, Belgium.

Wilks, Yorick. 2006. Artificial companions as a new kind of interface to the future Internet. Technical Report 13, Oxford Internet Institute. Available at: http://www.oii.ox.ac.uk/research/publications/RR13.pdf.

Winograd, Terry. 1976. *Understanding natural language*. Edinburgh University Press.

Witten, Ian H., and Eibe Frank. 2005. *Data mining: practical machine learning tools and techniques*. San Francisco: Morgan Kaufmann, 2nd edition.

Wonnacott, Thomas H., and Ronald J. Wonnacott. 1990. *Introductory statistics*. John Wiley and Sons, 5th edition.

Zender, Hendrik, and Geert-Jan M. Kruijff. 2007. Towards generating referring expressions in a mobile robot scenario. In *Language and robots: Proceedings from the symposium (LangRo'2007)*. Aveiro, Portugal.

Zender, Hendrik, Óscar Martínez-Mozos, Patric Jensfelt, Geert-Jan M. Kruijff, and Wolfram Burgard. 2008. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems* 56:493–502. Special issue "From sensors to human spatial concepts".