Interactive correction of speech recognition errors: implementation and evaluation for English and Swedish

Peter Ljunglöf, J. Magnus Kjellberg

Department of Computer Science and Engineering
University of Gothenburg and Chalmers University of Technology
peter.ljunglof@cse.gu.se, magnus.kjellberg@chalmers.se

1. Introduction

In the MUSTE project we explore how to make quick fixes to simple texts using as few interactions as possible (Ljunglöf, 2011). There are several situations where this could be useful, such as when you are driving (and don't have access to a keyboard), if your device is too small for a proper keyboard (such as a mobile phone), or if you have a communicative disability (e.g., cerebral palsy, visual impairment, or something else).

Assume that the user dictated a text message in their phone, and the speech recogniser got most of the message correct, but there were a few words that turned out slightly wrong. In the system that we envision, the user would point at the incorrect words, and the phone would then suggest possible substitutions based on phonological, syntactic and semantic properties. The suggestions for substitutions are presented in a menu from which the user can select the correct choice, or ask for a new menu of suggestions. The sentence can be further modified in small steps to finally reach the intended text.

At SLTC in 2016, we presented a very limited study to see if it would be interesting to investigate the approach further (Ljunglöf, 2016), and now we report on a larger-scale study that we conducted during spring 2018. The main goal of our study is to see if this kind of editing interface can be useful: how probable is it that the system suggests the intended correction, and what parameters are important for the system when calculating good suggestions? We present how the experiment system works, how we have evalutated its performance, and the evaluation results, for both English and Swedish speech recognition error correction.

2. Related work

Suhm et al. (2001) give an overview of strategies for speech error correction. One of the main strategies for correcting a misinterpreted word is to select from a list of alternatives, which is what we use in this project. The approach we are using is based on ideas from multimodal text editing (Ljunglöf, 2011), but we are using statistical models instead of grammars to suggest replacements. Liang et al. (2014; 2015) use a similar approach to ours, but they have a slightly more complicated interface with different editing operations, and they only evaluate Japanese. The Parakeet system (Vertanen and Kristensson, 2010) uses even more complex editing operations, making it possible to correct several errors at once, but on the other hand increases the cognitive burden on the user.

Previous evaluations of interactive speech input correction systems have mainly been performed on human subjects (Cuřín et al., 2011; Kumar et al., 2012; Suhm et al., 2001; Vertanen, 2006). In contrast, our evaluation is purely corpus- and lexicon-based and does not involve human subjects, which can be a promising complement to expensive evaluations on human subjects.

3. Implementation

When the selects the incorrect word(s), the system must come up with a reasonable list of substitution words. There are several possible approaches, more or less advanced. In this study we have chosen an approach in the middle when it comes to complexity.

3.1 Datasets

We use the following datasets in our system, for training the algorithms and for evalutation (see table 1):

Language model corpus: A large monolingual corpus for calculating n-gram frequences and language models. We used the English and Swedish Wikipedia,[1] containing approx. 1900m tokens (for English) and 370m tokens (for Swedish), respectively.

Parallel error correction corpus: A parallel corpus with speech recognition errors and their corrected counterparts. To create this corpus, we used a corpus for speech recognition training which consists of recorded utterances paired with gold-standard transcriptions. We automatically transcribed each recorded utterance with speech recogniser, and if the transcription differed from the gold-standard we added this transcription pair to our parallel corpus.

We created the English corpus from two open-source datasets, the VoxForge speech corpus[2] and Mozilla Common Voice,[3] totalling 270k recorded and transcribed utterances. We used the CMU Sphinx speech recognition toolkit[4] to transcribe the recordings. 32% of the utterances were recognised incorrectly, so our English parallel corpus contains 87k utterances.

The Swedish corpus is created from the dataset collected by Nordisk Språkteknologi (NST), freely available from the Norwegian Språkbanken,[5] containing 477k recorded and

[1] Wikipedia downloads, https://dumps.wikimedia.org
[2] VoxForge project, http://voxforge.org
[3] Mozilla Common Voice, https://voice.mozilla.org
[4] CMU Sphinx, http://cmusphinx.sourceforge.net
[5] NST database, https://www.nb.no/sprakbanken/repositorium

Dataset	English	Size	Swedish	Size
Language model corpus	English Wikipedia	1900m tokens	Swedish Wikipedia	390m tokens
Parallel error corpus (transcribed with)	VoxForge + Mozilla (CMU Sphinx)	171k errors	NST database (Google speech)	39k errors
Phonetic dictionary	CMU pronouncing dict.	123k entries	KTH phonetic dict.	938k entries

Table 1: Datasets used for training and evaluation.

	Utterances with errors	Substitutions involving at most 2 words on either side								
		total	1-0	2-0	0-1	0-2	1-1	2-1	1-2	2-2
English	87,307	76,009	6%	2%	8%	2%	40%	14%	12%	15%
Swedish	38,526	41,941	3%	<1%	6%	<1%	58%	17%	8%	8%

Table 2: Statistics for the parallel error corpora.

transcribed utterances. We transcribed 51k of the recordings using Google cloud speech recognition.[6] Google speech returns an n-best list, so we picked a transcription randomly from the 5 best candidates, to increase the number of incorrect transcriptions. Our final Swedish parallel corpus contains 39k utterances.

Table 2 shows the distribution of the different kinds of errors in the parallel corpora. In total there are 76k errors involving at most 2 words on either side for English, and 42k errors for Swedish. Most notable is that the by far most common error is a 1-1 word substitution.

Phonetic dictionary: A dictionary for converting between written text and their phonological representations. For English we used the CMU pronouncing dictionary,[7] containing 123k entries; and for Swedish we used the phonetic dictionary from the KTH Royal Institute of Technology,[8] containing 938k entries.

3.2 Workflow

After the system has recognised an utterance, it presents the sentence to the user. The user can then select a word, which is interpreted by the system as a request to replace the word with another word. The system does this by reordering a large internal dictionary, according to how probable it is that the new word is what the user originally intended when dictating the utterance. After reordering, the n topmost suggestions will be presented to the user, where n depends on the available space for presenting suggestions but in our evaluation we assume $n = 10$.

The system first uses an intitial filtering method to select the 10,000 most promising candidates from the starting dictionary. It is important that the initial filter is both efficient and selects good candidates, so we have tested three different methods for performing the first filter.

The candidates are reordered in a second phase. We use five different methods for calculating the probability that a dictionary word is a good substitution for the selected word. Logistic regression is used to combine the methods, and the candidate words are sorted by their final probability. For evaluation of logistic regression we used 10 times cross validation. The workflow is shown is figure 1.

[6]Google speech, https://cloud.google.com/speech-to-text
[7]CMU-dict., http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[8]KTH Swedish ASR models, https://www.speech.kth.se/asr

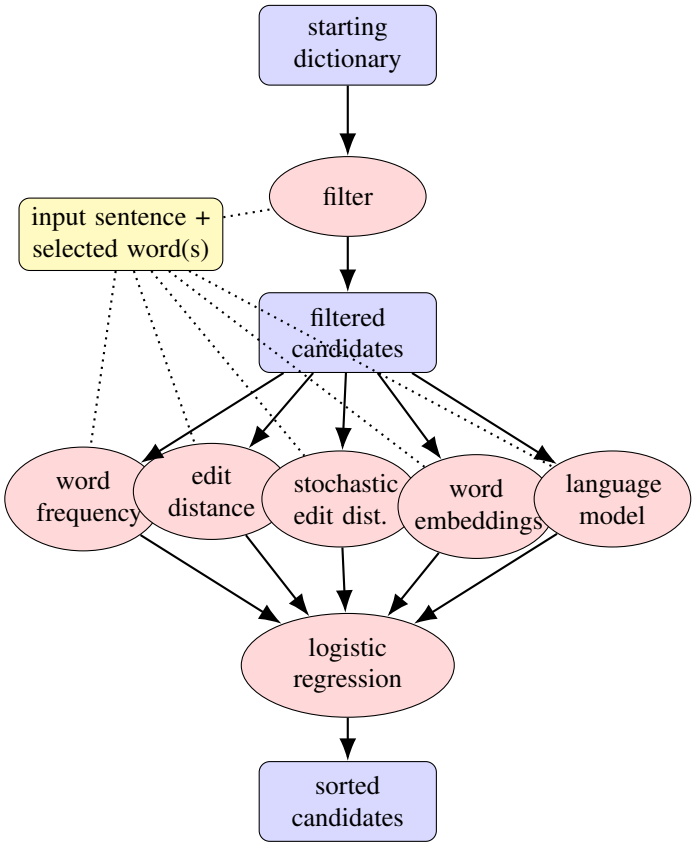

Figure 1: Workflow of the system

3.3 Models for error correction

To estimate the probability that a given dictionary word is the intended word, the system takes into account (1) how common the substitution is according to some corpus, (2) how similar the substitution is to the original word, and (3) how probable it is that the substitution blends in with the rest of the utterance. In our investigation we have implemented and tested five different methods for (1–3).

One of the intentions with our work is to investigate which methods work best for suggesting substitutions for misinterpreted words and phrases. We have implemented and evaluated the following five methods.

Word probability: All the substitution suggestions are taken from a large dictionary which is calculated from the

language model corpus. As an initial ranking of the words and phrases, we calculated unigram and bigram frequencies. To reduce the size of the database, we filtered out all bigrams with frequency less than 5. Finally we transformed all words into their phonological representations, using the dictionary. This resulted in an English frequency distribution for 123k unigrams and 3200k bigrams. For Swedish the corresponding figures are 185k unigrams and 1430k bigrams.

Word similarity: To measure the similarity between the selected word and the substitution, we use a phonological similarity score. This is measured by calculating the *Levenshtein edit distance* (Levenshtein, 1966) between the phonological transcriptions of the erroneous word and the correct word. Since the initial dictionary is so large, we need to be able to quickly filter the words that are close to the erroneous word. For this we pre-calculate a similarity index using SymSpell[9] (a similar algorithm is described by Bocek et al. (2007)). When building the similarity index, we have used a maximum edit distance of 5.

A slightly more advanced similarity measure is the *stochastic edit distance* which uses different weights for different phoneme pairs (Ristad and Yianilos, 1998). To train the weights we have used 10% of the parallel corpus. The implementation is much slower than SymSpell, so we can only perform this in the later reordering phase.

Utterance probability: We train a *KenLM language model* (Heafield, 2011) from the language model corpus. This language model is used for querying the syntactic probability of an utterance when replacing the selected word with an alternative.

Finally, we use *word2vec word embeddings* (Mikolov et al., 2013) trained from the language model corpus, as a semantic probability measure for the substituted utterance.

3.4 Replacing several words

It is possible that a selected word should be split in two or more shorter words (e.g., “awake” vs “a week”). It is also possible that two consecutive words should be merged into one (e.g., “camp fang” vs “campaign”), or even replaced with two other words (e.g., “her die” vs “heard I”). Our system is able to handle both 1- and 2-word substitutions, but the complexity increases when we want find a pair of words to suggest. E.g., the size of the English initial dictionary increases from 123k to 3200k, so the initial filtering method has to process more candidates, and the risk of suggesting bad substitutions increases. Nevertheless, we did conduct an initial study on some two-word substitutions.

4. Evaluation

We performed two evaluations: different methods for the first filtering phase, and different methods (and combinations) for the second reordering phase. These correspond to the pink ellipses in figure 1.

Our main evaluation has been on the most common corrections, where one word is replaced by one word. This is the 1-1 error type in table 2. An initial estimate tells that the accuracy of the other error types are worse, and our methods and workflow will probably need more thinking to improve correction of 1-2, 2-1 and 2-2 errors.

[9] SymSpell, https://github.com/wolfgarbe/SymSpell

Correct substitution...	English	Swedish
...is in intitial dictionary	99%	96%
...remains after first filter		
– SymSpell	84%	56%
– KenLM	67%	29%
– word2vec	82%	—

Table 3: Utterances where the correct suggestion remains after the first filtering phase.

4.1 First filter

We tried three different methods for filtering out the first 10k candidates: SymSpell, KenLM and word2vec. The evaluation was made on 1000 random utterances from the parallel corpus, and we measured for how many utterances, the correct suggestion still remained after the first filtering phase. As seen in table 3, SymSpell and word2vec both performed quite well for English, whereas KenLM fared worse. For more than 80% of the utterances, the correct candidate remained until the second phase. This suggests that the speech recognition errors are normally quite similar to the intended utterance, both with respect to phonology (SymSpell) and semantics (word2vec). We did not try to combine the three methods into a unified first filter, but that is of course a natural next step.

For Swedish the results are worse, and we have not done any investigation as to why this is. But one factor could be that the training corpora are smaller than their English counterpart. We did not have time to evaluate word2vec as first filter, for Swedish.

In addition we performed a limited evaluation of the error types 1-2, 2-1 and 2-2 for English. We only tested SymSpell, and the accuracy drops to 20–30% for these error types. We did not evaluate the second reordering phase for these error types.

4.2 Reordering the candidates

After filtering out the 10k most promising candidates, we reorder them. The n topmost candidates in this ordered list can then be presented to the user, where n depends on the available space for presentation. In this evaluation we assume that $n = 10$.

We tried all possible combinations of our five ranking methods. Table 4 shows the most important results: ALL means that we combine all five methods, $\neg m$ means that all methods except m are combined, and m means that we only used method m. The methods are abbreviated in the table: wf (word frequency), ss (SymSpell), sed (stochastic edit distance), klm (KenLM), and w2v (word2vec). The evaluation was made on 1000 random utterances from the parallel corpus, and we measured for how many utterances, the correct suggestion was among the top-10 suggestions after the second reordering phase.

Not surprisingly, the more methods we combine the better the accuracy. KenLM is the method which contributes the most, which is shown by the drop of accuracy when

	First filter (SymSpell)	The correct substitution is among the top-10 suggestions										
		ALL	¬wf	¬ss	¬sed	¬klm	¬w2v	wf	ss	sed	klm	w2v
English	84%	44%	45%	41%	44%	31%	42%	15%	23%	17%	36%	11%
Swedish	56%	38%	37%	32%	37%	35%	37%	8%	29%	9%	16%	2%

Table 4: Utterances where the correct suggestion is among the top-10 after the second sorting phase. The abbreviations are: wf (word frequency), ss (SymSpell), sed (stochastic edit distance), klm (KenLM), w2v (word2vec).

we leave it out, and the high accuracy when we only use KenLM. Stochastic edit distance seems to not be better than SymSpell, perhaps the weights are trained on too little data. Apart from that, it is difficult to draw conclusive conclusions. 44% of the English errors got the correct substitution among the top-10 candidates. For Swedish the results are in general 5–10 points lower, which probably partly has to do with smaller training data.

5. Discussion and future work

One conclusion to draw from this evaluation is that almost half of all 1-word speech recognition errors can be corrected using this touch-friendly method. With better ranking methods, better combination of the methods, and more training data, we are convinced that the accuracy can increase substantially.

Our next goal is to also increase the accuracy for 1-2, 2-1 and 2-2 substitutions, and perform a serious evaluation of those too. After that there are several possible paths:

- To improve the first filter by combining all methods, and perhaps add more methods – the important issue here is that the methods we use for first filtering must be very efficient.
- Investigate more ranking methods for the second phase, such as morphology or syntax. If available, context could be used for increasing the retrieval rate, e.g., topics and words from previous utterances and conversations. Bidirectional LSTM or other neural network architectures are also possible.
- Improve the datasets – if the main application is to correct text messages, we want to make use of a corpus of text messages.
- The pronunciation dictionaries can be improved, e.g., by using the recent CMU Sphinx G2P toolkit.[10]
- It would probably be very useful to use the internal information from the speech recogniser. Either the n-best list of results, or the internal states.

6. Acknowledgements

This research is funded by Chalmers ICT Area of Advance, and the Swedish Research Council (Vetenskapsrådet).

References

Thomas Bocek, Ela Hunt, and Burkhard Stiller. 2007. Fast similarity search in large dictionaries. Technical report, Department of Informatics, University of Zurich, April. http://fastss.csg.uzh.ch/.

Jan Cuřín, Martin Labský, Tomáš Macek, Jan Kleindienst, Holger Quast, Hoi Young, Ann Thyme-Gobbel, and Lars König. 2011. Dictating and editing short texts while driving: Distraction and task completion. In *AutomotiveUI 2011, 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, Salzburg, Austria.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of SMT 2011, the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, UK.

Anuj Kumar, Tim Paek, and Bongshin Lee. 2012. Voice typing: A new speech interaction model for dictation on touchscreen devices. In *Proceedings of CHI 2012, SIGCHI Conference on Human Factors in Computing Systems*, Austin, Texas, USA.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Yuan Liang, Koji Iwano, and Koichi Shinoda. 2014. Simple gesture-based error correction interface for smartphone speech recognition. In *Proceedings of Interspeech 2014*, Singapore.

Yuan Liang, Koji Iwano, and Koichi Shinoda. 2015. Error correction using long context match for smartphone speech recognition. *IEICE Transactions on Information and Systems*, E98–D(11):1932–1942.

Peter Ljunglöf. 2011. Editing syntax trees on the surface. In *Nodalida'11: 18th Nordic Conference of Computational Linguistics*, Rīga, Latvia.

Peter Ljunglöf. 2016. Towards interactive correction of speech recognition errors. In *SLTC'16, 6th Swedish Language Technology Conference*, Umeå, Sweden.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS'13, 26th International Conference on Neural Information Processing Systems*, pages 3111–3119.

Eric Ristad and Peter N. Yianilos. 1998. Learning string edit distance. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20, May.

Bernard Suhm, Brad Myers, and Alex Waibel. 2001. Multimodal error correction for speech user interfaces. *ACM Transactions on Computer-Human Interaction*, 8(1):60–98.

Keith Vertanen and Per Ola Kristensson. 2010. Intelligently aiding human-guided correction of speech recognition. In *Proceedings of AAAI'10, the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Keith Vertanen. 2006. Speech and speech recognition during dictation corrections. In *Proceedings of Interspeech 2006*, Pittsburgh, Pennsylvania, USA.

[10] G2P toolkit, https://github.com/cmusphinx/g2p-seq2seq