Towards interactive correction of speech recognition errors

Peter Ljunglöf

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

In this project we explore how to make quick fixes to simple texts using as few interactions as possible. 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).

The main contribution of this work is about improving the online interaction for a user who wants to correct speech recognition errors. The actual error correction algorithm that we have used – based on the Levenshtein edit distance – has not been in focus, and can probably be improved substantially.

1. Example scenario

One example scenario is if you want to send a text message while driving. You dictate your text to your phone and the phone's speech recogniser gets almost everything correct, there are only a few words that turn out slightly wrong ("will" vs. "well"), or one word is interpreted as two words ("proceed" vs. "process it"), or even two words which should be interpreted as two other ("her die" vs. "heard I"). It is also possible that you would need to insert a new word between to words, or to delete an extraneous word. So, perhaps the message that you intended to be like this,

"as you have **heard I will** now **proceed**"

instead turned out like this:

"as you have **her die well** now **process it**"

In this situation it would be great if you could fix the minor mistakes with just a few interactions, e.g., by pointing out the erroneous words or phrases ("her die", "well", "processed", or "it") and selecting the correct replacement ("heard I", "will", "proceed", or deleting the word) from the alternative words that the phone suggests. If the word you want is not among the suggestions, the system would suggest new alternatives to replace with.

This idea probably only works well if the recognised text is similar enough to the text that was intended. If the recogniser does too bad, we would have to resort to other approaches (e.g., to try to re-recognise the text again).

2. Related work

The approach we are using is similar to (Ljunglöf, 2011), but the main difference is that in this paper we are trying a statistical approach, as opposed to the grammar-based approach that was described in our earlier paper.

(Suhm et al., 2001) give a comprehensive overview of different strategies for error correction of speech recognition. The main strategies for correcting a misinterpreted word are (i) selecting from a list of alternatives, (ii) respeaking the word, (iii) spelling out loud, (iv) handwriting or other pen-based gestures, and (v) using a keyboard. In this paper we focus on (i), to select from a list of alternatives. According to Suhm et al, all strategies have their advantages and disadvantages, but the main disadvantage with (i) is that the accuracy tends to be low, i.e., the correct word seldom is among suggested list of words. However, their experiments are more than 15 years old, and both speech recognition quality and NLP techniques have become better since then.

(Liang et al., 2014a; Liang et al., 2014b) use a similar approach to ours, but they have a slightly more complicated interface, where the user makes different gestures on a touch-screen to indicate the position and type of all errors in the text, after which the system will suggest a new replacement text.

Previous evaluations of interactive speech input correction systems have all 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. Interaction with the system

The recognised sentence is a list of words, separated by whitespace. The goal of the interaction is to modify the sentence in small steps to finally reach the intended text. We provide the user with three interactive editing operations:

- The user can select (i.e., click, point or otherwise specify) a word in the text. The system interprets this as a request to either delete the word, or to replace it with another word (or a pair of words).
- Alternatively the user can select two words at once, which the system interprets as a request to replace both words at the same time.
- Finally the user can select a space between two words. This is interpreted as a request to insert a new word.

The system now tries to suggest new words to replace (or insert), and displays them in a menu. The user can select one of the suggestions, or they can ask the system to display another menu of replacement words.

4. Implementation

The main difficulty of the implementation is how to come up with a reasonable list of replacement words. There are several possible approaches, more or less advanced. In this initial attempt we have chosen a simplistic approach, to be able to perform an initial evaluation to see if the basic idea is feasible and worth exploring in more detail.

When a word is clicked, the system calculates a score for every word in a dictionary of possible replacements. The score of a replacement word is calculated from the frequency of the word, and its phonetic similarity to the original word.

The word frequencies are calculated from the unigram frequencies in the 3 million word Switchboard corpus, published as part of the Open American National Corpus.[1] We do not take the surrounding context into account when ranking, because the Switchboard corpus is too small to get useful bigram or trigram frequencies.

The phonetic similarity is measured by calculating the Levenshtein edit distance (Levenshtein, 1966) between the phonetic transcriptions of the source and the replacement words. The phonetic transcription is taken from the CMU pronouncing dictionary of the CMU Sphinx speech recognition engine,[2] containing more than 120 000 words.

Our approach for splitting a source word into two target words is done in a brute force manner: we split the source word at all possible locations and then apply the algorithm for single words twice.

5. Evaluation

We performed an initial evaluation to see if the basic idea would be worthwhile to explore further.

5.1 The evaluation corpus

To create a corpus of speech recognition errors, we used the open-source speech recognition training set collected by the VoxForge project.[3] This training set consists of 46 824 recorded utterances paired with gold-standard transcriptions. We used the CMU Sphinx speech recogniser to automatically transcribe each recorded utterance. 44% of the utterances were recognised incorrectly and we collected them into a parallel corpus of speech recognition errors which contains 20 556 utterances.

To measure how incorrect the recognition was, we used the Levenshtein character distance. The distribution of the "incorrectness" of the utterances is shown in table 1. As can be seen, the majority (52%) of the erroneous recognitions differ with at most 10 character edits.

This is encouraging for our idea of interactive editing of errors, since we believe that our approach works best for doing minor changes.

5.2 Experiments

Since our hypothesis is that click-and-select editing works best for correcting minor errors, we decided to do an initial evaluation of utterances with minor errors. From the evaluation corpus, we selected the 5551 utterances that had a character edit distance of 1–5.

Character distance	N:o utterances	
1–5	5551	27%
6–10	5127	25%
11–15	3495	17%
16–20	2557	12%
21–25	1586	8%
26–30	931	5%
31–85	1309	6%

Table 1: Distribution of errors in the evaluation corpus.

Error type	Correct	Possible	Difficult
Word deletion	14	–	–
Word insertion	5	2	3
Word replacement	54	13	8
Multi-word repl.	3	2	9
Total	76	17	20

Table 2: Distribution of suggested error corrections.

From this smaller evaluation set we randomly selected 100 random utterances, and manually annotated which replacements, deletions or insertions that should be done to correct the utterances. Some utterances require more than one correction, so in total there were 113 corrections that we evaluated. Of these corrections, 14 were deletions, 10 were insertions, 75 were single-word replacements, and 14 were multi-word replacements (such as "read I" to "red eye", or "fort" to "for it", or "camp fang" to "campaign").

For each of these 113 corrections, we asked the system to list the 50 highest ranked modifications. Then we manually checked if the correct modification was among these 50 suggestions. The results from this evaluation is shown in table 2.

The system suggested a correct modification in 76 of the 113 test cases. All deletions were correct, as were most of the single-word replacements, and half of the insertions. The multi-word replacements were more difficult however, which is not surprising.

We analysed the 37 failed cases in more detail, and came to the conclusion that 17 of them would probably have been treated correctly, if we had used more accurate bigram and trigram statistics from a larger corpus. Those 17 utterances contain very common sub-expressions such as "all rights reserved", "all the rest" and "passed through". The final 20 cases will probably require more work.

6. Discussion

Naturally a manual evaluation of only 113 examples is bound to give inconclusive results. However, the results that we got are very promising considering the simplistic algorithm we use for finding suggestions. For 76 of the utterances the system returns a correct modification among its list of suggestions. For another 17 utterances, we believe that there is a good chance that a slightly improved system will suggest the correct replacement.

[1] Open American National Corpus, `http://anc.org/`

[2] CMU Sphinx speech recognition toolkit, `http://cmusphinx.sourceforge.net/`

[3] VoxForge project, `http://voxforge.org/`

The evaluation is purely corpus- and lexicon-based and does not involve any human subjects, so it is possible to do an extensive automatic evaluation on the whole error corpus. However, such a fully automatic evaluation will not be able to distinguish between *possible* and *difficult* cases, as was shown in table 2.

As mentioned earlier, (Suhm et al., 2001) list five different correction strategies, and we have only focused on one of them – to select from a list of alternatives. The other strategies are of course still an option, and sometimes they are more useful. However, our strategy can be tried in parallel with other strategies – e.g., the system can present a list of options, and the user can either select the correct word, or speak (or spell) the correction out loud. In other words, using one correction strategy does not necessarily exclude using another strategy at the same time.

In some settings, displaying alternatives on a screen is not an option, e.g., when the user is visually impaired or when driving a car. However, by using a speech interface such as the voice cursor (Larsson et al., 2011a; Larsson et al., 2011b), a variant of our strategy could be very useful.

6.1 Future work

This is an initial report of an ongoing project. In the future we plan to try out several improvements of the selection and ranking algorithms for suggestions, such as:

- Use a distance measure that assigns different weights to more or less similar phonemes.
- Use a larger corpus for calculating word frequencies.
- Use context dependent features, such as bigram or trigram frequencies, so that the system can suggest words that make sense semantically.
- Use morphological analysis, so that the system can suggest grammatically correct inflections of the replacement words.
- Use part-of-speech tagging and grammatical analysis, so that the system can suggest words that make sense grammatically.

Furthermore, we plan to perform a fully automatic evaluation on the improved algorithms using the whole error corpus.

References

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.

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.

Staffan Larsson, Alexander Berman, and Jessica Villing. 2011a. Adding a speech cursor to a multimodal dialogue system. In *Proceedings of Interspeech 2011*, Florence, Italy.

Staffan Larsson, Alexander Berman, and Jessica Villing. 2011b. Multimodal menu-based dialogue with speech cursor in DICO II+. In *Proceedings of ACL-HLT'11: 49th Annual Meeting of the Association for Computational Linguistics, Systems Demonstrations*, Portland, Oregon.

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. 2014a. An efficient error correction interface for speech recognition on mobile touchscreen devices. In *Proceedings of SLT 2014, IEEE Spoken Language Technology Workshop*, South Lake Tahoe, Nevada, USA.

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

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

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