Mobile Texting: Can Post-ASR Correction Solve the Issues? An Experimental Study on Gain vs. Costs

Michael Feld[1] **Saeedeh Momtazi**[2] **Farina Freigang**[1] **Dietrich Klakow**[2] **Christian Müller**[1]

[1]German Research Center for Artificial Intelligence, Saarbrücken, Germany
firstname.lastname@dfki.de

[2]Spoken Language Systems, University of the Saarland, Germany
firstname.lastname@lsv.uni-saarland.de

ABSTRACT

The next big step in embedded, mobile speech recognition will be to allow completely free input as it is needed for messaging like SMS or email. However, unconstrained dictation remains error-prone, especially when the environment is noisy. In this paper, we compare different methods for improving a given free-text dictation system used to enter text-based messages in embedded mobile scenarios, where distraction, interaction cost, and hardware limitations enforce strict constraints over traditional scenarios. We present a corpus-based evaluation, measuring the trade-off between improvement of the word error rate versus the interaction steps that are required under various parameters. Results show that by post-processing the output of a "black box" speech recognizer (e.g. a web-based speech recognition service), a reduction of word error rate by 55% (10.3% abs.) can be obtained. For further error reduction, however, a richer representation of the original hypotheses (e.g. lattice) is necessary.

Author Keywords

Speech recognition, error correction, automotive, messaging

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation: Misc.

General Terms

Experimentation, Measurement, Performance

INTRODUCTION

Embedded, mobile speech recognition has evolved from simple voice commands ("*Call Anna*") to relatively natural speech input for certain domains ("*Where is the next Pizza place?*"). The next big step will be to allow completely free input as it is needed for messaging like SMS or email. Unconstrained dictation is the hardest form of automatic speech recognition (ASR) and remains error-prone, especially when the environment is suboptimal, because background noise is present or because the user can for some reason not fully concentrate on speaking. Unfortunately, in mobile scenarios this is mostly the case [5]. In the first place, this is an issue of optimizing the ASR engine, which can be done by taking enough of the right training material and by eliminating "nuisance attributes" from the recognition process. Following the post-ASR correction paradigm, we consider ASR as a black box, which cannot be optimized either because we already did our best or because we haven't got access to it. While post-correction can be applied in a number of mobile domains, the main application scenario we have in mind is dictating (and editing) text message while driving. Here, the black box scenario is very likely. Car manufacturers are in the process of moving from owned on-board ASR solutions for off-board third party services such as Vlingo [12]. Why? First of all, ASR development is costly. Then, innovation cycles of the product (car) need to be further decoupled from the much shorter cycles of value added components (such as infotainment). And finally, the bandwidth of mobile Internet has become wide enough to off-board services.

Obvious means for correcting errors that we know from personal computers at home will not be immediately applicable here, because in mobile situations, a number of constraints apply. On the other hand, correcting speech recognition errors by adding another speech recognition step bears the danger of ending up in cascading errors that frustrate the user [11]. [6] looked for modalities that are optimal for drivers using a combination of eye gaze and speech commands. However, independent from the ergonomics of the interaction, we should add some intelligence to the process, which helps reducing the number of necessary interaction steps as much as possible. In this article, we tackle the latter issue. Particularly, we address these questions: What factors do affect the performance of the error correction for the in-car domain? What methods can be used to improve it? Using a given ASR, what is the upper boundary of accuracy that can be achieved using these methods? And finally, how much user interaction is needed and how is it related to the performance?

RELATED WORK

Previous work on post-ASR correction using a single speech recognizer hypothesis focused on language models (LM) trained with statistical and other methods and relatively little training data. In all cases introduced in the following, ASR has been treated as a complete black box – the only clue given for the input string was the word sequence produced by the recognizer. The different approaches commonly employed LMs. [10] introduced a correction method using a back-off LM in order to allow domain specific tasks. [3] produced a LM consisting of two components: a word *n*-gram model accounts for lexical information and a maximum entropy language model (MELM) capturing higher level syntactic and semantic knowledge. In a similar study, [4] proposed a semantic-oriented approach. Here, lexico-semantic patterns (LSP) have been used that combined linguistic entries with semantic types in form of part-of-speech tags to capture constituents of utterance hypotheses. [7] developed a two-structured LM, comprised of syntactic-semantic and lexical information, as well as grammatical knowledge. Our model accounts for lexical information combined with either lexicographical or phonological similarities. Deletion and insertion errors where handled with the help of fertility models in the first three approaches, whereas our model so far includes deletions only. [13] target dialog systems with post-ASR confidence ratings based on both acoustic features, word lattice, and LM. [10] achieved a WER reduction of 14.9% using Sphinx-II for ASR and a LM trained on TRAINS-95 dialogue data for the transportation domain. However, the initial WER was rather high (42% respectively 35%). [3] improved an initial WER of 27% by 4%. They used LG-Elite and ByVoice ASR in an in-vehicle telematic domain. [4] used the same ASR and domain knowledge but also trained with the TRAINS-95 dialogue system and obtained a 6.28% WER-improvement. [7] employed data from the Saplen corpus to train different knowledge domains like fast food and Air Travel Information Service (ATIS). With these models they used the HTK ASR, which resulted in an overall 8.5% WER-improvement from a baseline of 23.88%.

POST-ASR ERROR CORRECTION APPROACHES

Fig. 1 illustrates the dialog system underlying this study. After activation, the user can speak a sentence, which is then displayed on a screen. Using one of several modalities such as touch, a central multi-functional device or short robust speech commands, an erroneous word can be selected, in which case a non-scrollable list of alternatives pops up. The user can select an alternative, choose the word to be deleted, or cancel the correction. The insertion of words is currently not implemented. Alternatively, on the main screen, the user can toggle between full sentence alternatives using left and right controls. When done, the user accepts the sentence. Given the "black box" scenario described above, one question immediately comes into mind: Where do the alternatives come from? The remainder of this paper is dedicated to this problem. Thereby, the term *interaction steps* refers to the number of individual input turns by the user, e.g. button presses. Hence, switching to the next sentence alternative is one interaction, selection a wrong word and then choosing an alternative is another two interactions.

Sentence Alternatives (DASA, WASA). Sentence alternatives, if provided by the ASR, can be used in multiple ways: for sentence-level correction (Directly Applied Sentence Alternatives, DASA) and to derive word alternatives (Word Alternatives based on Sentence Alternatives, WASA). Our WASA generator aligns all sentence alternatives with the (potentially incorrect) reference sentence on character-level using Needleman/Wunsch algorithm (using a score of 1 for match and -1 for mismatch and gaps) [9]. Then, words in the alternative are mapped to the words in the reference sentence depending on the characters they span in the alignment. It is possible that multiple words in the alternative are mapped to a single word (insertion) or that a single word in the reference

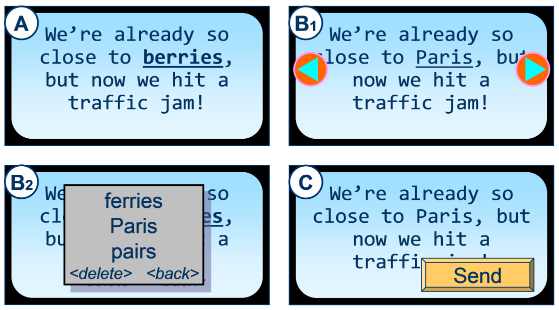
A
We're already so close to berries, but now we hit a traffic jam!
B1
We're already so close to Paris, but now we hit a traffic jam!
B2
ferries
Paris
pairs
<delete> <back>
C
We're already so close to Paris, but now we hit a traffic jam!
Send

Figure 1. Simplified view of the underlying dialog system. A) user dictates a sentence, B1) user selects alternative sentence, B2) user selects a word that is wrong and chooses an alternative word (using a minimal distraction modality which is not described here), C) user accepts the sentence by choosing "Send".

spanned multiple words in the source (deletion). The alternatives for a single word then correspond to the words from all sentence alternatives mapped to that word (minus duplicates). Word alternatives are ordered by the scores of the corresponding sentence alternatives from which they were obtained. If the number of alternatives was larger than the number offered to the user, the best n were displayed according to this metric. This method heavily relies on support from the recognizer and does not consider further domain-specific knowledge.

Language Model (LM). The second word alternative generator implemented does not require additional input beyond the sentence. Rather than that, it works solely on the context of the word and employs a pre-trained language model. The motivating idea is that the information about knowing which word is wrong, combined with the fact that the immediate context (the word before and after it) are then most likely correct, could be sufficient extra information for a statistical method. It is unclear, however, to what extend LM-based approaches depend on domain-specific training data. In our experiments described below, we therefore compared *in-domain* and *out-of-domain* LM variants. In any case, LM should be able to predict the words that can appear in the position of the wrong word reported by the user. This prediction is done based on the Bayes classifier scenario: First, the probability of each vocabulary term conditioned on the surrounding words is calculated. Typically, in the statistical language modeling, an n-gram model and specifically a trigram model is used to predict the present word based on the two previous already seen words. In our language model, two words in the immediate context are also used to calculate the probability of the present word. However, contrary to the normal recognition task which runs the LM simultaneously with the recognition, this LM is used after the recognition step and as a result, the model knows about all sentence's words, not only the predecessor words. Hence, the words before and after the present word are used for calculating the probability: $\hat{w}_i = \text{argmax}_w P(w|w_{i-1}, w_{i+1})$, where $\hat{w}_i$ is the word to be predicted as an alternative for the present word; w is the vocabulary word, and w_{i-1} and w_{i+1} are the previous and next words respectively. Similar to the normal trigram model, a smoothing technique is required to overcome the problem of unseen histories in which a bigram model is used as the back-off model for smoothing. Based on our experiments on the development set, using only the previous word for the bigram model performs better than using only the next word. Therefore, the previous word is considered as the backoff model while Kneser-Ney smoothing is applied. In the Bayes classifier, the best word which has the highest probability is selected as an alternative for the wrong word. In our model, since a list of alternative words is required, we further rank the vocabulary terms in the ascending format of this probability. Finally, top n words are selected as the best alternatives for the current position.

Lexicographical and Phonological Similarities. If the ASR has made an error, it is likely that the correct word is phonologically very close to the word reported as incorrect. Although the LM can have a good prediction based on the surrounding words, it does not take the actual word into account. This way, it is much more likely to suggest alternatives that are semantically or syntactically close to the word to be replaced, instead of those with similar acoustic features. While the correct word may still be among the top 50 alternatives sorted by LM probability, the limit for alternatives in our scenario require that the LM suggestions be re-ranked based on a metric that takes such similarity to the current word into account. We tested two such re-ranking schemes in our evaluation: A *lexicographical* and a *phonological* comparison. Both use a basic Levenshtein metric to sort alternatives by their distance to the original word. While the lexicographical method works on the written word in alphabetic characters, the phonological scheme works on the IPA phonetical representation, which is obtained using the *CMU Pronouncing Dictionary* [2].

EXPERIMENTAL SETUP

The input for the evaluation consists of a pre-processed text corpus (see below), which represents the message the user wants to send. In the next step, an ASR is applied which (at minimum) outputs one transcription for each input sentence. Each sentence is then processed like a human would do with a system as it is described here, until either it is completely correct or the remaining errors cannot be fixed using the options provided. Modules generating the alternatives have access to the sentence in its current state, and – if available – any extra information from the recognizer. They do not know the correct version of a respective sentence, but they may be trained on external data prior to the experiment. For word-level correction, the word alternatives module is queried for options for each wrong word. If the correct solution is among them, it is selected. The size of the alternatives list is one of the main parameters in the following analysis. The two measures applied are *word error rate* (WER) of the original and processed sentences, as well as *interaction steps* (IS). It is important to note that interactions are counted even if they don't lead to an improvement, because the system has to expose the same level of knowledge as the user: If the user has to click a wrong word to see whether the correct alternative is available, the system should too.

The corpus is a custom collection of publicly available material from a popular microblogging service ("MB corpus"). 51 users were selected with a total of 63,841 short sentences (average 8.6 words/sentence and 4 characters/word) in English, with a special preference for automotive, traffic and traveling topics (some users indeed seemed to be writing while driving). All sentences were post-processed by regular expressions designed to remove artifacts and transform the output into a format resembling dictated speech. E.g., small numbers were changed to their written form, common abbreviations were spelled out, etc. A subset of 940 sentences was selected manually. Three native American English speakers read each sentence. The eval set consists of 482 sentences while the remainder of the corpus is used as training (63,383) and development (458) set for the *in-domain* LM. The development set is used to study the smoothing parameters. The training data consisted of 452,137 word tokens and 27,361 word types. Additionally, we trained an *Upper-Boundary* model directly on the eval set used as recognizer input. It measures the upper boundary performance that can be achieved with any LM training data based on the recognizer output.

The out-of-domain model was trained on the *Blog06* corpus created by the University of Glasgow [8]. It is a collection of homepages and permalinks from blogs monitored over an 11 week period from Dec 2005 to Feb 2006. The collection contains 3,215,171 pages. Since blog documents are user created, we found that this dataset is close to our application yet larger and more generic than the in-domain LM. We tested two recognizers: the *SAPI 5.3*-based recognizer included in Windows 7 (R_{WIN}) and the end-user version of *Dragon Naturally Speaking 10* (R_{DRAG}). R_{WIN} provides sentence alternatives and scores, while the R_{DRAG} only provides the transcription. Neither of the two recognizers was adapted to the speaker beforehand, i.e. both used generic US English models. Additionally, we created a "pseudo" recognizer (R_{PSEUD}) that takes the correct transcript and replaces random words in a controlled fashion. We tested a number of parameters in different combinations. The first goal was to find out how much the initial error rate could be improved if the number of IS would not play a role (i.e. find an upper boundary for the method). The second goal was to see how the error rate would change if a tolerance level with respect to the maximum number of IS was defined.

RESULTS

With the R_{WIN} recognizer, our baseline is the initial WER of 18.4% (and stddev of 21.8). Different degrees of improvement can be achieved using Directly Applied Sentence Alternatives (DASA) correction, Word Alternatives based on Sentence Alternatives (WASA), and LM. For each method, the number of top alternatives that are checked for the correct word is varied, starting from a low number that requires few IS because it can fit on a single screen without the user having to page or scroll, and moving towards a number (we chose 50) that models optimal performance. In addition, the maximum number of alternatives generated for LM and ASR was 50 respectively 20. For the LM method, we also compared the different training sets and ranking schemes. We omitted ranking tests for WASA because we can assume that all of the ASR's alternatives are already ordered w.r.t. similarity to the word to be corrected. On the lowest, but also least costly alternatives setting (5), the improvement in WER absolute percent is 4.8 for DASA, 5.6 for WASA, and 1.8 for LM using *in-domain* training. The corresponding numbers for the highest setting (50) are 9.4, 20.2, and 3.7. While the LM achieves a notable improvement, its "upper boundary" (6.5) still falls behind the methods that are based on extra information from the ASR. On the other hand, the performance only slightly decreases for the out-of-domain model (3.2 with 50 alternatives). In the cases where the number of displayed alternatives is smaller than the number of available alternatives (e.g. 20 vs. 50), a lexical or phonological re-ranking was performed. The phonolgical ranking almost always outperformed the lexical ranking, although the difference was not decisive ($< 1\%$). These results, as well as all others in this section, are statistically significant based on a two-tailed paired t-test at the level of *p-value* < 0.01.

The R_{DRAG} ASR has a lower initial WER, and so the effects are less visible, but still significant. Because it offers no sentence alternatives, only the LM-based correction method could be applied. Apart from that, it shows the same trends at 1.7 (in-domain), 4.2 (upper-boundary) and 1.51 (out-of-domain). The R_{PSEUD} recognizer was created to see if there was a relationship between ASR errors and difficulties of the two alternative generators. It is very likely that "real" recognizers make errors with seldom or difficult words, which are tough cases for the correction as well. Our observations appear to confirm this, as the same LM achieves a greater improvement on the artificial (evenly distributed) errors.

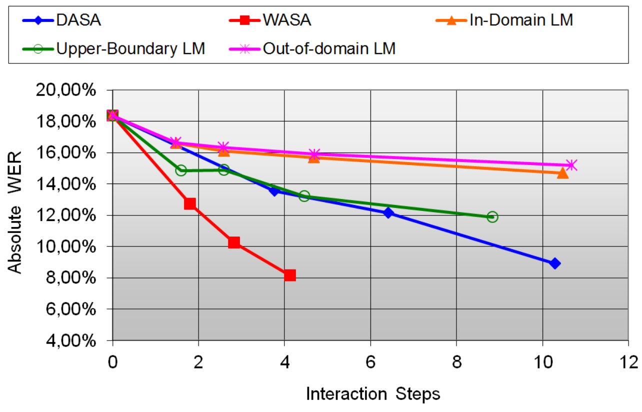

Figure 2. Tradeoff between interaction cost and correction performance. Marks correspond to the number of alternatives provided, which are 0**,** 5**,** 10**,** 20**, and** 50 **(the upper limit is imposed by the ASR used).**

Fig. 2 shows the IS' respective cost corresponding to the major experiments using R_{WIN}. For the LM conditions, the phonological ranking was selected. Recommendations in published standards [1] indicate that approximately five word alternatives can be displayed legibly on a 7" screen, so we counted an additional IS for each "page switch" the user would have needed to do. From the chart it can be seen what the improvement in WER for a given maximum number of IS may be. For example, if we allow four IS, the LM will provide a performance improvement of approximately 2.5% abs. However, if the ASR provides sentence alternatives, the improvement would be 10% abs. if WASA is used to generate word alternatives from them, and it requires approx. 4.1 IS.

CONCLUSIONS

This study presented a number of options how to deal with speech recognition errors in a scenario where user interactions are particularly costly and ASR configuration is limited. The approaches were compared in terms of performance, but also in terms of required interaction and with respect to their ASR feature requirements. The experiment suggests that an advanced ASR provides both the best overall accuracy as well as a good interaction/performance trade-off, but it also shows that post-correction matters: A purely LM-based approach, which requires no ASR insight, can still result in a solid performance increase of 3.7% abs. WER. While this is the biggest improvement the LM could achieve in our test, it requires approx. ten user interactions per sentence. If she commits only 2.6 interactions, which is more realistic in driving situations, the same method results in a gain of 2.6% absolute.

REFERENCES

1. ISO/TR 16352: Road vehicles – Ergonomic aspects of in-vehicle presentation for transport information and control systems, 2005.
2. Carnegie Mellon University. The CMU Pronouncing Dictionary, 2007.
3. Jeong, M., Jung, S., and Lee, G. G. Speech recognition error correction using maximum entropy language model. In *Proc. of INTERSPEECH 2004* (2004).
4. Jeong, M., Kim, B., and Lee, G. G. Semantic-oriented error correction for spoken query processing. In *Automatic Speech Recognition and Understanding (IEEE Workshop)* (2003), 156–161.
5. Jiang, X., Ma, M., and Chen, C. Speech recognition on mobile devices. In *Proc. WMMP 2008*, Springer (2010).
6. Kern, D., Mahr, A., Castronovo, S., and Müller, C. Making use of drivers' glances onto the screen for explicit gaze-based interaction. In *Proc. of AutomotiveUI-2010*, ACM (2010), 110–113.
7. López-Cózar, R., and Callejas, Z. ASR post-correction for spoken dialogue systems based on semantic, syntactic, lexical and contextual information. *Speech Communication 50*, 8-9 (2008), 745–766.
8. Macdonald, C., and Ounis, I. The trec blogs06 collection: Creating and analysing a blog test collection. Techn. rep., U.Glasgow, 2006.
9. Needleman, S. B., and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol. 48*, 3 (1970).
10. Rigger, E., and Allen, J. F. A fertility channel model for post-correction of continuous speech rec. In *Proc. of ICSLP* (1996), 897–900.
11. Vertanen, K., and Kristensson, P. Parakeet: A continous speech recognition system for mobile touch-screen devices. In *Proceedings of IUI 2009* (2009).
12. Vlingo Corporation. Vlingo Mobile, 2010.
13. Zhang, R., and Rudnicky, A. I. Word level confidence annotation using combinations of features. In *Proc. Eurospeech 2001* (2001).