# Choosing Useful Word Alternates for Automatic Speech Recognition Correction Interfaces

*David Harwath*[1]*, Alexander Gruenstein*[2]*, and Ian McGraw*[2]

[1]Massachusetts Institute of Technology
[2]Google, Inc.

## Abstract

Speech recognition is an increasingly important input modality, especially for mobile computing. Because errors are unavoidable in real applications, efficient correction methods can greatly enhance the user experience. In this paper we study a reranking and classification strategy for choosing word alternates to display to the user in the framework of a tap-to-correct interface. By employing a logistic regression model to estimate the probability that an alternate will offer a useful correction to the user, we can significantly reduce the average length of the alternates lists generated with no reduction in the number of words they are able to correct.

## 1. Introduction

In the last several years, automatic speech recognition has seen an unprecedented level of adoption, especially by users of mobile devices. A likely contributor to the attractiveness of speech input is the cumbersome nature of virtual keyboards. A typical person's typing speed averages between 50 and 100 words per minute on a full keyboard, but only 10 words per minute on a virtual keyboard [1]. Conversely, humans are able to dictate at approximately 102 words per minute, but after taking into account the time required for correcting speech recognition errors, this rate can drop to less than 10 words per minute [2]. Speech input has the potential to be vastly more efficient than virtual keyboard input, but thus far has faced a significant scourge in recognition errors. While better recognition accuracy has gone a long way, errors are unavoidable in any practical system which makes efficient error correction absolutely necessary.

Many error correction strategies for speech recognition have been studied in the past, and fall into several categories. Re-entry methods require the user to repeat some or all of the misrecognized utterance, or to fall back to keyboard entry. Other approaches take advantage of competing recognition hypotheses in the form of an N-best list, word lattice, or word confusion network (WCN). Because WCNs offer a highly compressed and often more interpretable view of a word lattice, they are a popular representation for spoken language understanding [3, 4] as well as speech recognition correction [5, 6, 2]. In particular, [5] demonstrated that touch-screen devices are well-suited for WCNs since the user need only tap the correct words on the screen to correct a spoken input. A more lightweight representation similar to a WCN is the alternates list, in which a user can first indicate the misrecognized words in a hypotheses and then be prompted to select from a list of candidates to replace the misrecognized text. In [2], the authors performed a user study investigating speech correction approaches on a tablet PC. Their results indicated very high user satisfaction for both alternates lists as well as re-dictation. They also found that users would typically opt to use the alternates lists first, falling back on re-dictation when the correct word was not in the alternates list.

Because they are well suited for mobile touch-screen devices and generally satisfying to users, alternates lists are a promising method of speech correction. It is important, however, to show the correct word or phrase in the list, while minimizing the number of incorrect corrections displayed to the user. In this paper, we investigate a strategy for automatic re-ranking and selection of hypotheses with which to populate an alternates list. In [7], the author explores the corrective potential of WCN lists as a function of their maximum allowable size. The WCNs were originally derived from word lattices, although users were given the option to replace words in the WCN with morphologically similar variants. What differentiates our approach from [7] is the use of a discriminative classifier applied directly to the alternates lists generated for individual words and phrases.

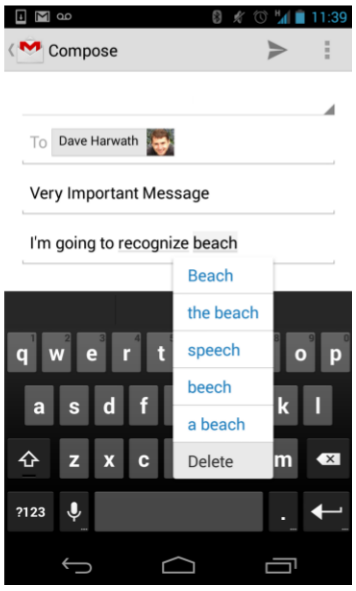
This work was performed while the first author was an intern at Google.

## 2. Correction Using Alternates

We illustrate the word alternates correction interface in Figure 1. When a user taps on an incorrect word on the screen, a list of alternates appears below the word and the user can select a replacement by tapping on the desired alternate in the list. What differentiates an alternates list from a word confusion network is the fact that alternates allow a user to replace words at the short phrase level as well as the word level. In our experiments, we find these candidate alternates using an N-best list of recognition hypotheses with timing information. Let $w$ be a word in the recognition hypothesis aligned to the audio interval $[t_0^w, t_1^w]$. To find alternates for $w$, we scan the N-best list entries for words approximately aligned to the same audio interval. To quantify the amount of overlap between $w$ and some other word $v$ aligned to the audio interval $[t_0^v, t_1^v]$, we compute

$$\frac{\min(t_1^v, t_1^w) - \max(t_0^v, t_0^w)}{t_1^w - t_0^w} \tag{1}$$

When this overlap exceeds a threshold (0.1 in all of our experiments), $v$ is considered a candidate alternate for $w$, assuming that $v$ does not already appear in the alternates list for $w$ (i.e. duplicates are not allowed.) Note that multiple consecutive words in a single N-best entry may overlap $w$, in which case the phrase formed by those words is considered a single candidate alternate for $w$. Additionally, this technique can be easily generalized to handle consecutive words in the recognition hypothesis by concatenating their alignment intervals. We do this for every sequence of words in the hypothesis whose total length does not exceed 10 characters in order to generate alternates for short phrases.



To Dave Harwath

Very Important Message

I'm going to recognize beach

Figure 1: An example of an alternates-based correction interface on a Nexus 4 smartphone running Android version 4.3

When offering word or phrase alternates to a user, our goals are twofold. First, the alternates should enable the user to correct as many word errors as possible. Second, the alternates lists should be concise. Not only is screen space limited on mobile devices, but forcing a user to read through long alternates lists is a burden which should be avoided. In practice, there is a tradeoff between these goals since increasing the size of the alternates lists tends to increase the odds that the proper correction is contained within the list.

Given an erroneous speech recognition hypothesis and a set of alternates, the problem of predicting which of those alternates will be useful to the user and which will not can be viewed as a binary classification problem. We define an alternate to be useful if it has the potential to correct word errors in a recognition hypothesis. Consider a contrived example in which a user spoke "mary had a little lamb", but the utterance was misrecognized as "mary had a little yam". Assume the two words "ham" and "lamb" appear as alternates for the word "yam". An oracle with knowledge of the reference transcription attempting to minimize the number of word errors using the available alternates would choose to replace "yam" with "lamb". We therefore consider "lamb" to be a useful alternate, as opposed to "ham", which is not useful. We employ this oracle technique to our training and evaluation sets, assigning useful alternates a "1" label and useless alternates a "0" label. At runtime the oracle is clearly not available, so we resort to estimating the probability that each alternate would be used by the oracle to correct one or more word errors. Because of its simplicity and ease of training, we choose to use logistic regression to predict this probability. Logistic regression also supports highly flexible feature spaces, and can easily handle combinations of binary and continuous features.

## 3. Features

For each candidate alternate generated for a recognition hypothesis, we extract a set of features to be used for regression. We explore:

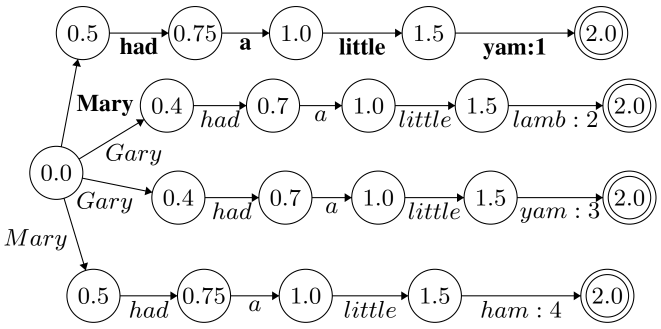


Figure 2: An N-best lattice with timing information displaying the 4 best paths for a contrived utterance. The utterance is 2 seconds long and contains the words "Mary had a little lamb" misrecognized as "Mary had a little yam". Timing information is tracked by the state labels, and the N-best position is reflected as the cost attached to the final word in each path. The 1-best path (which would be displayed to the user as the recognition hypothesis) is shown in bold.

*N-best list position features.* The first feature we consider reflects how far down in the N-best list the alternate appears. This feature takes on as its value the index of the N-best entry in which the alternate is first found. The deeper we must delve in the N-best list to find the alternate, the larger this feature becomes. We also include multiple binary features to reflect when an alternate appears in specific N-best entries. For example, when an alternate appears in the 2nd best hypothesis, the corresponding binary feature takes on a "1" value; if this same alternate does not appear in the 3rd best hypothesis, then the feature corresponding to the 3rd best path takes on a "0" value. We include these binary features for the 2nd, 3rd, 4th, and 5th best paths, as well as an extra feature representing all paths beyond the 5th best. Because an alternate may appear in multiple N-best paths, several of these binary features may be active at the same time. The last N-best position-based feature we employ is the N-best rank. The N-best rank of an alternate reflects the number of other unique alternates appearing before it in the N-best list.

Suppose we have the N-best list shown in Figure 2. Now suppose we wish to generate alternates for the word "yam" in the 1-best hypothesis. The N-best depth feature of the alternate "lamb" would be 2, and the N-best rank feature of the alternate would be 1 since no other alternates appear before it in the N-best list. The N-best depth feature of the alternate "ham" would be 4 since it first appears in the 4th best hypothesis, but the N-best rank feature of "ham" would be 2 since the alternate "lamb" appears before it. We do not count "yam" from the 3rd best hypothesis as an alternate, since it is the very string in the 1-best hypothesis we are attempting to replace.

*Posterior features.* Many speech recognition word confidence classifiers rely on posterior probabilities, so we consider those features here. Given a word or phrase $v$ in the 1-best hypothesis and a candidate alternate, $w$, we estimate the posterior probabilities for $v$ and $w$ using the likelihoods of the N-best list entries. Assuming that the likelihoods of the entries in the N-best list have been normalized to sum to 1, we accumulate the likelihoods of all the N-best paths which contain $v$ at the same time-aligned position as $v$ in the 1-best hypothesis. We repeat this computation for all paths containing $w$ at the same time aligned position to estimate a posterior probability of the alternate $w$. Both the posterior for $w$ as well as the posterior for $v$ are included as features for the alternate $w$.

*Text distance-based features*. We also compute features relating to the text of the alternate $w$ and the text of the 1-best word or phrase $v$. The absolute string lengths of $w$ and $v$ are included as features for the alternate $w$, but we also compute several features representing the string distance between $w$ and $v$. The first of these is the simple Levinshtein distance with equal insertion, substitution, and deletion costs. The last features capture the relative lengths of $w$ and $v$. When an alternate $w$ is longer than the string $v$ it replaces, the relative overshoot is given by

$$overshoot = \frac{len(w) - len(v)}{len(v)}, \qquad (2)$$

where $len(\cdot)$ represents string length, and the overshoot is taken to be 0 when $len(w) < len(v)$. We also use the relative undershoot,

$$undershoot = \frac{len(v) - len(w)}{len(v)} \qquad (3)$$

when $len(w) < len(v)$. In the case that $len(w) > len(v)$, the undershoot is taken to be 0.

## 4. Experimental Conditions

In our experiments, we aim to investigate the tradeoff between the fraction of word errors correctable using alternates, and the average length of each list of alternates. We can trade between these quantities by adjusting the accept threshold imposed upon the estimate of the posterior probability that an alternate can correct an error. As this threshold is raised we would expect fewer alternates to be selected, reducing the expected alternate list length but also the chance that a useful alternate appears in said list. We make several experimental assumptions which we describe in detail here. First, we assume that when an utterance is correctly recognized, a user will not tap on any words to see an alternates list. Conversely, when an utterance is completely misrecognized, the user is more likely to re-speak the query than attempt to correct every single word using alternates. Therefore, we limit our training and evaluation to utterances which possess a small number of errors. In our experimental setup, approximately 70% of the utterances in our training and testing sets were correctly recognized, approximately 25% had between 1 and 3 word errors, and approximately 5% had more than 3 word errors. We filtered these sets and only selected utterances with 1 to 3 errors for training and evaluation. Furthermore, we only wish to evaluate alternates generated for the erroneous portions of each utterance. Even though alternates will almost certainly appear in the N-best list for words in the hypothesis that were correctly recognized, they will not be displayed to the user unless those words are selected for replacement, which is unlikely. Therefore, our training and testing data is entirely comprised of alternates lists generated for misrecognized words in utterances with less than 3 total word errors. Our training set consists of English voice search queries sampled from anonymized logs of voice input on Android smartphones. For testing purposes, we also use anonymized voice search logs, but spread across English, French, Italian, German, and Spanish. Table 1 describes each of these datasets in more detail.

To generate the N-best recognition hypotheses used in our experiments, we use a state-of-the-art, deep neural network-based speech recognition system. The details of the speech recognizer recipe are described in [8]. After decoding the training and evaluation data, alternates are generated for each utterance using an N-best list. The 0/1 regression target labels are derived using the oracle word error scorer described in Section 2, and features are computed for each alternate. To train the logistic regression models, we selected a balanced training set of 6,706 positive and 6,706 negative examples of alternates from the English voice search training set. We trained one model with all of the N-best position, posterior probability, and text distance features, as well as a model which did not utilize the posterior probability features or the Levinshtein distance feature. We compare these models to a baseline classifier which utilizes only the N-best depth feature. This baseline, parameterized by a single integer $n$, accepts only the alternates which appear in the top $n$ best hypotheses in the N-best list. In our experiments, we cap N to 30, and also restrict the maximum alternates list length to 5.

| Language | # Utterances | # Words | # Word Errors |
|---|---|---|---|
| English (Train) | 18308 | 94621 | 29093 |
| English (Test) | 6615 | 30810 | 10476 |
| French | 2062 | 6087 | 3223 |
| Italian | 1094 | 3189 | 1701 |
| German | 1549 | 3676 | 2422 |
| Spanish | 1097 | 3342 | 1671 |

Table 1: Statistics of the various datasets used in our experiments. These statistics reflect the utterances actually used for training and testing in our experiments (i.e. after filtering out correctly recognized utterances and utterances with more than 3 word errors)

For evaluation, we sweep an accept threshold from 0 to 1 over the regression estimate of the probability that a given alternate will be useful for correction. Likewise, for the N-best depth baseline, we sweep the depth parameter $n$, retaining only the alternates that appear in the top $n$ N-best paths for a given utterance. For each threshold setting, we examine the alternates list generated for each erroneous word or phrase in the evaluation data. We compute the fraction of those word errors correctable using the available alternates, and also compute the average number of alternates appearing in each list.

## 5. Results

The tradeoff between these quantities on our English voice search testing set is shown in Figure 3. The blue asterisks mark the discrete operating points available to the N-best depth-based classifier, while the red and black curves respectively trace the operating points of the full and simplified logistic regression models. Since the N-best depth is just one of the features utilized by the models, we would expect both models to outperform the simple N-best depth-based classifier, which is indeed the case. For a given average alternates list size, the alternates selected using the logistic regression models are always able to correct a larger number of word errors. As the accept threshold is lowered, the average alternates list length grows larger, yet the number of word errors correctable with alternates saturates around 30%. A reasonable operating point might be to set the accept threshold high enough to make 29% of the word errors correctable. The motivation behind this choice is that it makes the majority of the good alternates available to the user, and increasing the average list length further tends to add alternates with no corrective potential. At this operating point, the lists produced by the regression models are on average 2.6 alternates

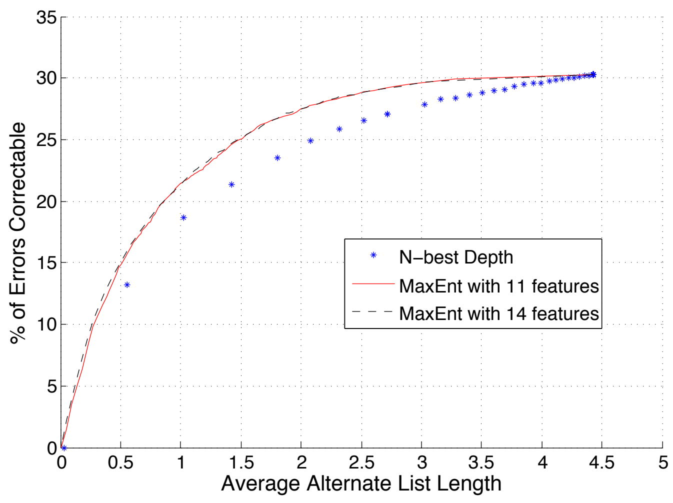


Figure 3: Correctability / List Length Tradeoff for English Voice Search. The 14-feature and 11-feature maximum entropy models perform very similarly, and their curves significantly overlap.

long, while the lists produced by the N-best thresholding classifier are on average 3.6 alternates long. This is a significant reduction in average list length which comes at no cost. As can be seen in Figure 3, the posterior probability and Levinshtein distance features do not aid in classification. This is somewhat surprising given the fact that lattice posteriors are often used as features in ASR confidence modules, although it is possible that the N-best position and rank features make the posterior probability features redundant. This result is good news from an implementation standpoint, however, since it means that in practice these features need not be computed. Our experiments also show that the N-best lists saturate rather quickly and contain enough alternates to correct approximately 30% of all errors.

We also investigated the application of our alternates classifier to non-English languages, namely French, Italian, German, and Spanish. Figure 4 displays the tradeoff between average alternates list length and correctability for each of the non-English test sets. The same classifier trained on the English data was used on all four of these languages, and in every case gave improvements comparable to the English results. This is a pleasing, since it means that a single classifier can be trained and applied to multiple languages with good results, without requiring training data in each target language.

## 6. Conclusion

In this paper, we have presented a method for selecting word alternates from an N-best list for the purpose of speech recognition error correction. The method employs a simple logistic regression model to estimate the probability that an alternate will offer a useful correction to the user. In practice, the average length of the alternates lists generated was reduced by 28% at a reasonable operating point with no loss in the corrective power of the alternates. Furthermore, we demonstrated that a the logistic regression model trained on English data was able to generalize to other languages with good results.

Although we demonstrated a respectable reduction in the average alternates list length, future work should investigate

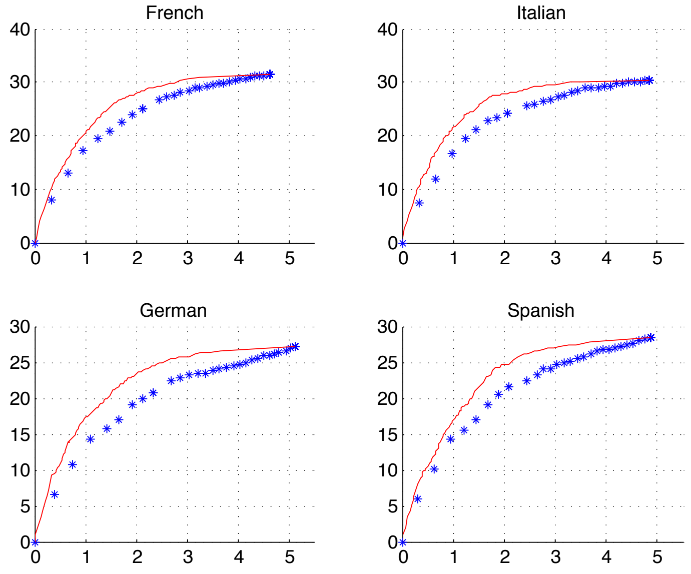


Figure 4: Correctability / List Length Tradeoff for the non-English languages investigated. Average alternate list length is on the horizontal axis, while the fraction of errors correctable is on the vertical axis. The red line traces the performance of the simplified (11 feature) Maximum Entropy classifier, while the blue asterisks represent the N-best thresholding baseline.

methods of populating the alternates lists from additional sources of information. In our experiments, our word lattices contained only enough words to correct 30% of all word errors. Using a wider search beam during decoding would provide larger lattices from which to draw more N-best entries, at the cost of increased decoding time. Another method would be to employ dictionary access based on phonetic distance from a hypothesis word in order to recover near-homophones which may not appear in the decoding lattice. Regardless of the methods used to generate additional alternates, reducing the number of unhelpful alternates drawn from the N-best list provides extra room for these additional alternates; a single classifier could even be trained to select alternates from this pool of different sources.

## 7. References

[1] A. Cockburn, A. and A. Siresena, "Evaluating mobile text entry with the Fastap keypad," in *British Computer Society Conference on Human Computer Interaction*, England, 2003.

[2] K. Larson and D. Mowatt, "Speech error correction: the story of the alternates list," in *International Journal of Speech Technology* vol. 6, no. 2, 2003.

[3] J. Feng and S. Bangalore, "Effects of word confusion networks on voice search," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computation Linguistics*, 2009, pp. 238-245.

[4] D. Hakkani-Tur, F. Bechet, G. Riccardi, and G. Tur, "Beyond ASR 1-best: using word confusion networks in spoken language understanding," in *Computer Speech and Language* 20(4):495-514.

[5] K. Vertanen and P.O. Kristensson, "Parakeet: a continuous speech recognition system for mobile touch-screen devices," in *IUI '09: Proceedings of the 14th International Conference on Intelligent User Interfaces.* ACM, 2009, pp. 237-426.

[6] J. Ogata and M. Goto, "Speech repair: quick error correction just

by using selection operation for speech input interfaces," in *Proceedings of Interspeech*, Lisbon, Portugal, 2005.

[7] K. Vertanen, "Efficient correction interfaces for speech recognition," PhD thesis, University of Cambridge, 2009.

[8] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proceedings of Interspeech*, Portland, Oregon, 2012.