Estimating Word-Stability During Incremental Speech Recognition

Ian McGraw[1], *Alexander Gruenstein*[2]

[1]Massachusetts Institute of Technology
[2]Google

imcgraw@mit.edu, alexgru@google.com

Abstract

Many speech user interfaces can be improved by incrementally displaying or interpreting a speech recognizer's current best path as a user speaks. This gives rise to a problem of *instability*, whereby the best path may change frequently, particularly with respect to the words most recently spoken. Introducing a lag between the audio most recently processed and the portion of the best path shown to the user can lead to more usable incremental results. In the ideal case, the lag introduced would vary to recover exactly the longest stable prefix of the best path. In this paper, we introduce a framework for estimating a stability statistic for each word, and explore the tradeoff of stability and lag by thresholding stability statistics estimated using a variety of features.

1. Introduction

Incremental speech recognition, in which partial recognition results are streamed from a speech recognizer while the user is talking, has been of interest in the speech community for decades [1]. In general, incremental results have been shown to make a spoken language system feel more natural and responsive to the user [2]. Unlike systems that wait for silence to return a result, incremental recognition trails the audio as closely as possible, much like closed captions on television. Indeed, automatic video captioning is one of incremental speech recognition's many applications [3].

The use of incremental speech recognition results has been explored in academia, e.g. [4], and is now appearing in commercial applications. Since version 2.2, Android has included a type-by-voice feature. In version 4.0, Android's latest release, incremental results allow users to watch the recognition unfold as they are speaking. Incremental results have also proven useful outside of speech recognition. For instance, Google Instant Search, which searches with every keystroke as users type, saves on average 2 seconds per search [5]. Incremental speech recognition could result in similar time savings in some tasks. The user could see words, and they could be interpreted, before the end-pointer has even fired.

With incremental speech recognition results, however, a new issue of *stability* arises. Simply put, the recognizer can "change its mind" about which words it thinks have been spoken, causing the streamed incremental results to contradict one another, especially with respect to the most recently decoded portions of the utterance. One can stabilize incremental results either by choosing *when* to emit them, or by emitting just a prefix deemed stable. Selfridge *et. al* describe three methods of choosing when to emit incremental recognition results and use logistic regression to predict the stability of the entire incremental result [6]. Baumann et. al explicitly modify incremental recognition results according to duration-related features [7] and assess the stability of the newly formed incremental result (often prefixes of the best path in the decoder).

In contrast to previous research, we take inspiration from the practice of assigning confidence statistics to each word in a hypothesis. Instead of estimating the probability that a word is correct, however, the stability statistic estimates the probability that every word in an incremental hypothesis up to and including that word will remain the same in all future incremental hypotheses. Although certainly related, these two measures are clearly different. It is easy to imagine a word in a partial utterance that would receive a low confidence score, but is very stable due, perhaps, to subsequent portions of the lattice.

Our approach to stability estimation uses logistic regression to compute a statistic for each prefix of an incremental result. This can be viewed as a combining the strengths of [6] and [7] into a framework that allows a client of the recognizer to retain fine grained control over the handling of instability. In particular, since every prefix of an incremental recognition result is labeled with a stability statistic at the word level, a dictation application, for example, might employ a simple threshold to display only the portion of the result deemed stable.

In the remainder of this paper, we explicitly define the stability problem and discuss features, such as right-context, age, and word-posteriors that might be used to estimate a stability statistic. We then compare three feature sets on both Voice Search and Voice Input data from Android phones. We analyze the tradeoff each application might make between the stability of a prefix and the amount it lags behind the decoder. We see that, for Voice Search, a combination outperforms a single feature alone.

2. Stability

We illustrate the stability problem using a contrived example shown in Figure 1. Here we are showing the distinct incremental results which appear at particular frames (shown in parentheses). In actuality, we sample the best path from the decoder more frequently, however, in our example this would just produce a continuous stream of identical incremental results between the distinct ones shown.

If we suppose the purpose of an incremental result is to inform the user as early as possible of the words in the final speech recognition hypothesis, then it becomes apparent from this example, that some incremental results are better than others. The hypotheses at frames (10) and (20) are entirely misleading, while the one at frame (230) is mostly correct except for the final word.

The finer details of the definition of stability for a particular

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

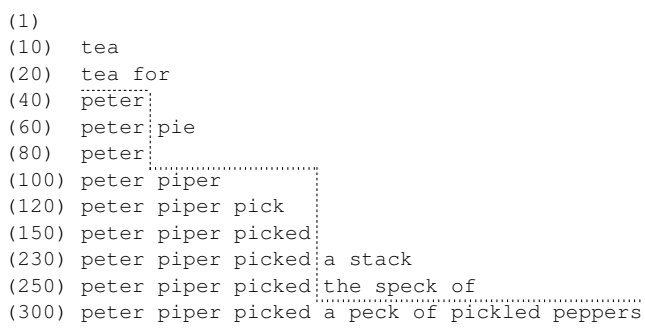

Figure 1: Unstable incremental results, with the best possible prefixes lightly outlined.

prefix of an incremental result is a matter of choice. One might prefer a definition which compares the prefix only to the next distinct incremental result, or perhaps only to the final result. In our work, however, we define a prefix to be stable only if all future incremental results have the same prefix. Thus, the word "a" does *not* end a stable prefix at frame (230), whereas the word "peter" at (60) does. Although typically sub-word prefixes are not considered under our definition of stability, we make an exception when an entire word is a prefix of another word in the subsequent incremental result. Thus, the word "pick" at (120) also ends a stable prefix (in this case the entire incremental result.)

Given a set of incremental results, like those shown in the example, it is possible to compute ideal prefixes with respect to our stability criterion. In Figure 1, these prefixes are lightly outlined. Immediately apparent is the fact that, even with ideally prefixed incremental results, there is likely to be some degree of lag between the time the word was actually spoken and when it becomes stable on the decoder's best path (assuming it was recognized correctly.)

Of course, the ideal prefixes are not known a priori. We therefore resort to estimating a stability statistic, which can be used to recover prefixes that are likely to be stable from each incremental result. A reasonable stability estimate for each prefix would allow a client of the recognizer to operate on longer prefixes when instability can be tolerated and shorter prefixes when high stability is required. A shorter prefix, however, would incur a longer lag. This tradeoff between the lag behind the decoder and the stability of the incremental results becomes the center of our analysis in later sections.

3. Features

Similar to word confidence modules, our word stability module can handle arbitrary features extracted during decoding. Given a word w_t occurring in an incremental result in our training corpus, we extract features from the relevant utterance up to time t to predict the stability of the prefix ending in w_t. The feature vector we create, $f(w_t)$, can be as small as a single feature, or contain thousands of features. In this section, we describe some of the features we have explored.

We begin with two duration-related features, similar to those examined in [7]. The **right-context**, c_w, of a word w measures how close (in seconds) to the end of the incremental result the word resides in an alignment of the incremental result. A slightly more subtle duration-based feature is that of **age**, a_w, which we define to be the amount of time that a prefix has survived on the best path without changing as more audio has been decoded. Unlike the related "message smoothing" feature described in [7], our framework does not allow the incremental results to conflict with the decoder's best path.

Data Set	WER	# Utts.	Words
Training	17.2	27,327	6.4
Voice Input	13.2	49,649	5.6
Voice Search	17.1	27,273	3.2

Table 1: Word error rate, number of utterances, and words per utterance for the training and test sets.

In addition to duration-based features, one can draw features from the the word lattice built during decoding. In this paper, we experiment with word-level **posteriors**. Often such posteriors are used in confidence modules [8], however, here we must compute word posteriors for partial results. The probability that a word is correct, p_w, given the evidence seen so far can be computed from a lattice representing current hypotheses and scores up to the last frame decoded. Ultimately any number of features can be drawn from this lattice and incorporated into our framework to predict stability.

Another property of our search space we may wish to capture is its size. For example, the immortal nodes described and explored in [1] and [6] indicate points in the lattice where all paths converge. In systems with high-perplexity, a more general notion of search-space size might be useful. With cost efficiency of primary concern, we take a simple measurement, s_w, of our search space size at the exact moment w appears in terms of the number of arcs currently active divided by the maximum number of arcs seen so far in the decoding process. This feature attempts to capture roughly how large our search space is relative to how large it has been in the past.

With enough data, we can split our feature space by the very **words** we are trying to compute stability over. More generally, we can have indicator functions signify when a word w is found in one of K categories, $\mathbb{1}_{W_k}(w)$. The choice of how to define the sets W_k can be made in a number of ways. In this work, we give each of the 1000 most common words in our training set its own category, and thus indicator function, allowing the remainder to fall into a set of less frequently seen words. This catch-all set consisted of words seen fewer than 54 times, and represented about one quarter of our training set.

We have, of course, left many possible features unexplored, however, we turn now to a few **interaction** terms. For each of the word indicators above, we have defined a feature representing its age $a_w \times \mathbb{1}_{W_k}(w)$. Including both the interaction term and the indicator function for each set gives us 2002 word-related features. We also incorporate $a_w \times p_w$ and $a_w \times s_w$ into our most ambitious model.

4. Stability Estimation

Since transcriptions are not required to ascertain true stability, we can potentially use an arbitrarily large amount of training data. We preferred, however, to use a training set with well understood properties and so we chose a set of transcribed utterances in English sampled from anonymized logs of Voice Search and from the Voice Input capability on Android phones. Voice Input [9] utterances consist primarily of short message dictation, e.g. SMS messages. Voice Search [10] utterances typically contain spoken queries, and tend to be shorter. Our test sets consisted of utterances from anonymized logs of the same two applications, sampled from a different time period. We report results on Voice Search and Voice Input utterances

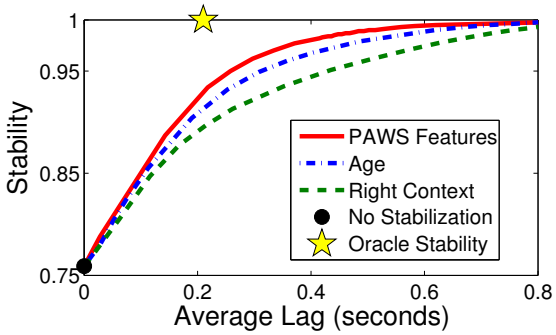

Figure 2: Stability/Lag tradeoff for Voice Search.

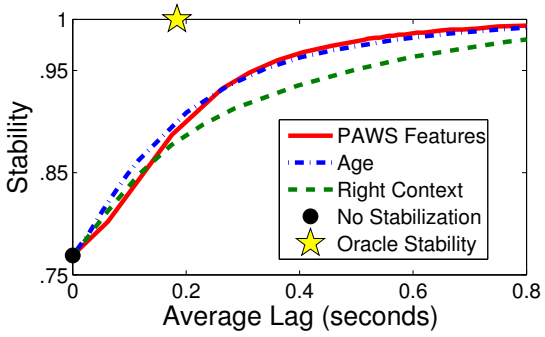

Figure 3: Stability/Lag tradeoff for Voice Input.

separately, given the differences between two applications. Table 1 provides details about the datasets, such as word error rate (WER) and average utterance length.

The incremental recognition results generated by our training set contained 444,328 individual words (where a word is considered the same across partial results if it appears in the same position). We decoded using a 4-gram language model with a vocabulary of approximately 1 million words, and a tied-state triphone GMM-based HMM acoustic model trained using ML and boosted-MMI objective functions. We use PLP-cepstral coefficients and LDA.

We use a regression to estimate the probability that a prefix of an incremental result is stable given features associated with its final word. Due to the ubiquity of its training software and the simplicity of its parametrization, we present results here using logistic regression. Given data of the form $\langle w_t, y_t \rangle$, we populate a feature vector, $f(w_t)$, of size M to train a set of parameters $\beta \in \mathbb{R}^M$. A single frame was randomly sampled from each of the 444,328 hypothesized words in the incremental results of our training set, and features $f(w_t)$ were computed at that frame. The binary response y_t was recorded to represent the true stability of the prefix ending in w_t. With our parameters trained, we can predict our stability statistic $s = logit^{-1}(\beta f(w_t))$.

5. Analysis

We perform our analysis from the perspective of a client of the speech recognizer. We believe that a simple threshold, whereby the client makes use of the longest prefix with a stability of at least $t \in [0, 1]$, will be one of the more common operations performed on stability. One caveat with thresholding the raw statistic, however, is that spurious deletions may occur if a stability happens to oscillate around the threshold. Also, given our definition of stability, it makes sense to *ensure* that stabilities decrease from left to right in a given incremental result. We analyze stability under both of these monotonicity constraints.

Regardless of how they are chosen, it is clear that using prefixes of incremental results will introduce a lag between the current frame being processed by the decoder and the last frame represented in the prefix. In a dictation application, for example, this manifests itself in the delay between speaking a word and seeing it onscreen. It is this tradeoff between lag and stability, that we wish to analyze here.

Numerous metrics to analyze the tradeoff between stability and lag have been thoroughly explored in [7]. We have opted for the somewhat simpler approach of comparing a single metric of stability to a single metric of lag. To compute the average number of seconds of audio ignored by a given stability threshold, we examine a partial alignment at every frame, and sum together the time corresponding to words that do not meet the threshold. We then divide by the duration of the corpus to retrieve the average amount of lag the threshold introduces.

It is important to remember that t defines the minimum stability that the client finds acceptable, so the stability felt by the end user may actually be significantly higher. To obtain a notion of stability at a given threshold, we therefore compute the fraction of frames which contain a stable prefix when thresholded using t. We can then use t as an operating parameter and sweep out a plot of the average lag experienced against the fraction of the time a stable prefix is active. Under this corpus-level notion of stability both our test sets were already around 75% stable when the entire best path is always used. It should be noted that analyzing distinct results would not look as rosy. Our metrics examine every frame, however, to provide a fair comparison across all our feature sets.

In Figures 2 and 3, we show the stability improvement on Voice Search and Voice Input data as a function of lag. The oracle point represents the stability and lag of the ideal set of prefixes of the incremental results in the test set. The three curves represent stability threshold sweeps on regressions learned from three different feature sets. The first feature set is simply c_w, the right-context of a word. The second is again a single feature a_w, age. It should be noted that we allowed intercept terms for the regression on each of these single-feature sets. Clearly, age is more indicative of stability than right-context. This result is intuitive, however, in that even if a word has a large amount of right-context, a young age implies that it has changed in the recent past and may change again.

Given that age was such a strong feature, we decided to combine it with the other features in our final set. In addition to the interaction terms $a_w \times p_w$ and $a_w \times s_w$, we include the 2002 word-based features, mentioned in the previous section. We refer to this choice of features as the PAWS feature set since it includes features based on posteriors, age, words, and a search statistic.

On the Voice Search test set, this configuration achieves operating points closest to the oracle. On the Voice Input set, however, the gains are reduced to almost nothing. We believe the differences are largely due to utterance length. We have chosen features that target a single word to predict the stability of an entire prefix. While the monotonicity constraints we impose do propagate low stabilities within a partial, we believe that features geared towards explicitly capturing longer dependencies would be worth exploring to improve performance on long utterances. We could introduce language model scores or other

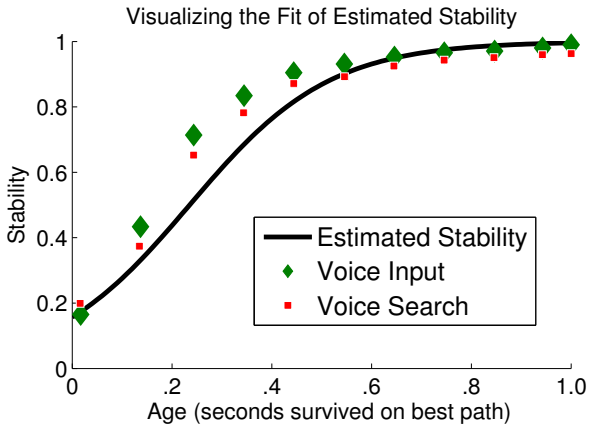

Figure 4: A sigmoid learned for the age feature accompanied by the actual fraction of stable utterances in a set of evenly distributed bins.

cross-word statistics easily into the logistic regression framework.

While the tradeoff analysis provides a clean picture of how the stability of a corpus of thresholded prefixes relates to the lag introduced, it does not explicitly show how accurate the stabilities themselves are. To visualize this, we have chosen to plot the regression learned using the age feature against the true distribution of the data in the test sets in Figure 4. Prefixes from the test sets, sampled in the same manner as the training set, were binned according to their age. The mean age of each evenly spaced bin was plotted against the fraction of prefixes that were stable in that bin. We can see from the binned test data that the learned curve does a reasonable, though not perfect, job of fitting the true data. It appears that we are slightly underestimating probabilities below about .9, but then the estimates become more accurate.

One way to achieve a more accurate estimate would be to bin the feature-space itself. Indeed, there are numerous feature representations that we have yet to explore. Fortunately, the logistic regression framework can flexibly handle a large number of features while robustly handling data sparsity issues should they occur. Finally, the framework can be trained efficiently with distributed techniques, making it possible to handle extremely large datasets [11].

6. Conclusions

We have presented a framework for the computation of stability of incremental speech recognition results modeled after the word confidence module typical in speech recognition systems. Rather than computing a statistic regarding word correctness, however, the stability statistic measures the likelihood that a word ends a prefix which remains stable through the remaining incremental results. Note that while we have focused on assigning stabilities to prefixes delineated by word boundaries, the methodology developed in this paper could be applied to sub-word units as well, opening up the possibility of displaying words themselves incrementally as they are spoken.

We use logistic regression to model stability, reducing the problem to one of feature selection, and present two single-feature models and one multi-feature model. Perhaps surprisingly, the simple "age" feature, indicating the length of time an incremental result has been on the best path, performs remarkably well. We show further gains using a feature set that incorporates thousands of word-related features. While more exploration needs to be done to translate these gains to dictation-like test sets, the framework makes it straightforward to experiment with arbitrary features.

The analysis we have performed in this paper examines lag in terms of decoded frames of the incoming audio. While this isolates stability-lag tradeoff, it ignores the effects that the network or computational efficiency can have on lag. These concerns might be addressed by revisiting either the features themselves or the rate at which they are sampled. We leave these experiments, however, to future work.

It is our belief that faster, more stable incremental results will translate into a more natural user experience. Indeed, the improvements shown in this paper are easily noticeable at the level of an individual utterance. When aggregated across a large user-base, however, the impact of saving a few hundred milliseconds becomes truly significant. Perhaps equally important is the fact that our framework allows each spoken language application to determine how best to make use of the incremental speech recognition results.

7. Acknowledgements

We would like to thank Brian Strope for helping us configure the data sets and making useful suggestions along the way.

8. References

[1] P. Brown, J. Spohrer, P. Hochschild, and J. Baker, "Partial Traceback and Dynamic Programming," in *Proc. of ICASSP*, 1982.

[2] G. Aist, J. Allen, E. Campana, C. G. Gallo, S. Stoness, M. Swift, and M. K. Tanenhaus, "Incremental dialogue system faster than and preferred to its nonincremental counterpart," in *Proc. of the 29th Annual Meeting of the Cognitive Science Society*, 2007.

[3] M. Saraclar, M. Riley, E. Bocchieri, and V. Goffin, "Towards automatic closed captioning: low latency real time broadcast news transcription," in *Proc. of ICSLP*, 2002.

[4] S. Kanthak, S. Molau, A. Sixtus, R. Schlüter, and H. Ney, "The RWTH large vocabulary speech recognition system for spontaneous speech," in *Proc of the Konvens*, 2000, pp. 249–254.

[5] About google instant. Accessed: March, 2012. [Online]. Available: http://www.google.com/insidesearch/instant-about.html

[6] E. Selfridge, I. Arizmendi, P. Heeman, and J. Williams, "Stability and accuracy in incremental speech recognition," in *Proc. of SIGDIAL*, 2011.

[7] T. Baumann, M. Atterer, and D. Schlangen, "Assessing and Improving the Performance of Speech Recognition for Incremental Systems," in *Proc. of NAACL-HLT*, 2009.

[8] G. Evermann and P. C. Woodland, "Large vocabulary decoding and confidence estimation using word posterior probabilities," in *Proc. of ICASSP*, 2000.

[9] B. Ballinger, C. Allauzen, A. Gruenstein, and J. Schalkwyk, "On-demand language model interpolation for mobile speech input," in *Proc. of Interspeech*, 2010.

[10] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Garrett, and B. Strope, "Google search by voice: A case study," in *Advances in Speech Recognition: Mobile Environments, Call Centers, and Clinics*. Springer, 2010.

[11] G. Mann, R. Mcdonald, M. Mohri, N. Silberman, and D. D. Walker, "Efficient large-scale distributed training of conditional maximum entropy models," in *In Advances in Neural Information Processing Systems*, 2009.