INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Rescoring-aware Beam Search for Reduced Search Errors in Contextual Automatic Speech Recognition

Ian Williams, Petar Aleksic

Google Inc.
{iantw,apetar}@google.com

Abstract

Using context in automatic speech recognition allows the recognition system to dynamically task-adapt and bring gains to a broad variety of use-cases. An important mechanism of context-inclusion is on-the-fly rescoring of hypotheses with contextual language model content available only in real-time.

In systems where rescoring occurs on the lattice during its construction as part of beam search decoding, hypotheses eligible for rescoring may be missed due to pruning. This can happen for many reasons: the language model and rescoring model may assign significantly different scores, there may be a lot of noise in the utterance, or word prefixes with a high out-degree may necessitate aggressive pruning to keep the search tractable. This results in misrecognitions when contextually-relevant hypotheses are pruned before rescoring, even if a contextual rescoring model favors those hypotheses by a large margin.

We present a technique to adapt the beam search algorithm to preserve hypotheses when they may benefit from rescoring. We show that this technique significantly reduces the number of search pruning errors on rescorable hypotheses, without a significant increase in the search space size. This technique makes it feasible to use one base language model, but still achieve high-accuracy speech recognition results in all contexts.

Index Terms: speech recognition, decoding algorithms

1. Introduction

In automatic speech recognition (ASR) systems, using contextual information to modify speech recognition language models in real-time allows a general speech recognition engine to perform well on a variety of tasks. For example, a user issuing a search query via voice may refer to entities that are geographically relevant, or a voice-interaction device may be in a state which expects an utterance to be one of a particular set of phrases [1] [2]. In voice search, we have previously shown [3] improved recognition from including context based on users' search history and context based on entities that appear more frequently for users in particular geographic regions.

One example of a misrecognition fixed by geographic context is "Save me a pair of used cars" becoming "St. Mary Parish used cars" for a user near St. Mary Parish, Louisiana.

On-the-fly rescoring can be used to include contextual information in the base language model (LM). It changes costs assigned by the base LM to specific n-grams during beam search. For example, if we are adapting the recognizer for a media control device which is currently playing music, we may choose to override the base cost for the bi-gram "stop playing" to something significantly lower. This rescoring happens on the lattice as words are emitted by the decoder, and the updated costs are used as the search continues [4].

This solution has a problem; search hypotheses that fall outside some threshold of the most-likely hypothesis must be pruned in order to keep the search time- and space-efficient. Since the rescoring model is not applied until a word has been recognized, a good hypothesis can be pruned at the intra-word level [4]. Longer, less-likely words which share a prefix with shorter, more-likely words suffer from this issue in particular. For example, "armadillo" could sound very much like "arm a . . ." initially, especially if the user pauses or slows after the initial "arm" sound. "Armadillo" could be pruned before rescoring is applied. This is possible with any beam search, but is exacerbated for words whose contextual cost is much lower than the base LM cost.

In this paper we describe a solution to this problem by introducing a modified beam-search algorithm which is able to look ahead from intra-word hypotheses and allow a higher pruning threshold in cases where a rescored word is reachable from that hypothesis.

We organize our paper as follows. In section 2 we describe a contextual ASR system and how context is consumed by the decoder. In section 3 we describe pruning, our modification to the pruning algorithm, and how rescored word reachability is determined. In section 4 we describe our test sets and our experimental results. Lastly, in section 5 we summarize our findings and list future work areas.

2. Contextual ASR System

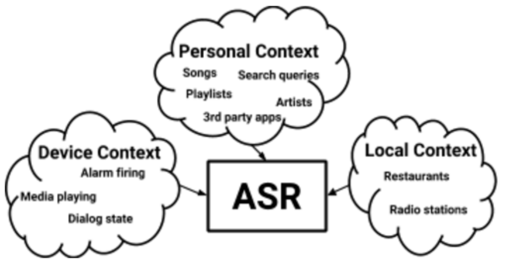

Figure 1: *Contextual ASR system with examples of injected contexts.*

Our contextual ASR system (Figure 1) handles many different domains. We adapt the same base recognizer to handle voice search, device-specific command recognition, mobile application voice control, and third-party commands, among others. Within these domains we introduce strong and weak contexts. A strong context corresponds to, for example, a user being prompted to respond to a question by choosing from one of a few fixed responses. A weak context corresponds to a light decrease in cost for a subset of the n-grams in the base LM. We previously described a contextual ASR system that utilizes user history and geography-based weak contexts [3].

Our decoder functions similarly to the one described in [5]. It uses weighted finite state transducers (WFSTs) based on the OpenFST library [6]. Introducing context in the speech recognition system is handled by a variety of techniques, including dynamic language model classes [7], out-of-vocabulary (OOV) word injection [1], and on-the-fly LM rescoring for both strong and weak contexts [3].

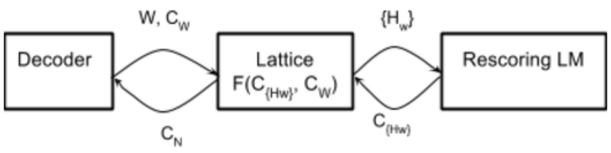

Figure 2: *On-the-fly rescoring system.*

An on-the-fly LM is applied in the following fashion (Figure 2): when an arc emitting a word w is traversed and the lattice is updated to reflect this traversal, we find the word histories H_w associated with that search hypothesis and perform a lookup in the rescoring LM for H_w. If the rescoring LM assigns a new cost C_{Hw}, an interpolation function F is applied on the base LM cost C_W and the rescoring LM cost C_{Hw}. The new cost $C_N = F(C_w, C_{Hw})$ is updated accordingly [4]. In our use-cases, we only increase the likelihood of n-grams. If an n-gram is assigned a higher cost in the rescoring model than the base LM, we do not rescore.

3. Algorithm Design

Our decoder graph is similar to that described in [8]. The search graph is constructed dynamically via WFST composition of the C, L, and G WFSTs. C encodes the context-dependency of the acoustic model triphones, mapping acoustic model outputs to phonemes. L encodes the lexicon, and maps phonemes to the words in the recognizer vocabulary, and G encodes the base language model. The complete $C \circ L \circ G$ WFST is far too large to fit in memory, hence $C \circ L$ is composed ahead of time, and dynamic WFST composition with G is performed lazily, as needed during decoding.

3.1. Pruning thresholds

The pruning threshold defines the maximum allowable cost difference between the lowest cost hypothesis in the search graph and one under consideration for expansion. If this value is exceeded, the hypothesis under consideration is not expanded. The pruning threshold T_i applied in the current frame i to an arc leaving an active state is a function of an empirically-chosen beam size B, an empirically-chosen maximum arc threshold M, and the number of arcs expanded in the previous search frame N_{i-1}. Let $A_{i-1,j}$ be the jth arc expansion in the $(i-1)$th frame and $C(A_{i-1,j})$ be its cost difference from the best-scoring hypothesis. Further assume that $A_{i-1,j}$ is cost-ordered, so that $C(A_{i-1,j-1}) \leq C(A_{i-1,j}), \forall j > 0$. Then,

$$T_i = \begin{cases} B & \text{if } N_{i-1} < M \text{ or } i = 0 \\ min(B, C(A_{i-1,M})) & \text{otherwise} \end{cases}$$

We introduce a second threshold T^R which follows the same rules as above, with $T^R >= T$, and T^R applied when an arc $A_{i,j}$ lies along a path which can reach a rescored word and when $C(A) > T$. T^R is computed similarly to T. We choose new $B^R > B$ and M^R. A^R_{i-1} denotes only the arcs that were retained by rescoring and N^R_{i-1} be the number of arcs retained by rescoring. Then,

$$T^R_i = \begin{cases} B^R & \text{if } N^R_{i-1} < M^R \text{ or } i = 0 \\ min(B^R, C(A^R_{i-1,M^R})) & \text{otherwise} \end{cases}$$

3.2. Reachability Analysis

In order to determine if an arc from one state to another lies along a rescorable path and apply the second threshold T^R, we used the following technique:

1. Retrieve the word histories for the hypothesis
2. Lookup the rescorable words given the word histories
3. Lookup the states on paths reaching those words
4. Check if the state hypothesis in question is contained within the result of 3.

Step one is trivial and handled already by the token passing algorithm used in the search, which means every hypothesis has a pointer to the lattice from which we can find the word histories the hypothesis represents [5]. Step two is solved by requiring that the rescoring LM be able to enumerate the words it will rescore given a word history. As our rescoring LMs are encoded as WFSTs where each state represents a word history [4], this is trivial and requires only linear time with respect to the number of rescored words for that word history.

Steps three and four are more complex since the search graph is composed dynamically as needed, and there is no stable mapping from a word to a set of states reaching that word. Instead we use the WFST composition of the $(C \circ L)$ and G to determine the mapping from a $C \circ L \circ G$ state to the underlying $C \circ L$ state. Next we need to determine whether that $C \circ L$ state reaches the word in question. All that remains is to have a mapping $M : W \mapsto \{S_W\}$, where W is a word in the recognizer vocabulary V and $\{S_W\}$ is the set of $C \circ L$ states reaching W.

M can be precomputed when the $C \circ L$ is constructed, and loaded into memory for recognition. We store the states of each set $\{S_W\}$ concatenated together in an array $[\{S_0\}\{S_1\}...\{S_{|V|}\}]$, where $|V|$ is the size of V. We also create a lookup array which is indexed by W and gives the interval start and end for W in the $\{S_W\}$ array. The size of this data structure is given by $\sum_{W \in V} |\{S_W\}| + |V|$. We can reduce $|\{S_W\}|$ by not including those states that reach a number of words larger than some threshold. An appropriate threshold drastically reduces the size of M without much penalty since a $C \circ L$ state which reaches a large number of words will not have much LM weight pushed forward [8].

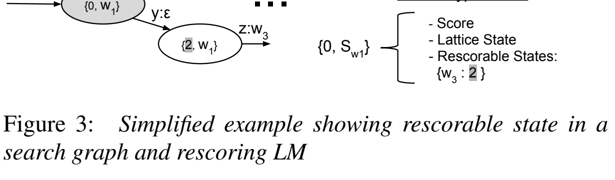

Figure 3: *Simplified example showing rescorable state in a search graph and rescoring LM*

Figure 3 shows a simplified section of a search graph, along with an active state and an applicable rescoring LM. The highlighted state $\{0, w_1\}$ is active and has been reached via an arc emitting word w_1. The active hypothesis has its associated score, a pointer to the lattice state for its word histories (in this case just w_1), and the states which lead to rescorable words. The rescoring LM state 1 matches this state's word history of w_1, hence word w_3 is eligible for rescore. As we can observe on the arc leaving state $\{2, w_1\}$, word w_3 is reachable via state $\{2, w_1\}$. When the decoder considers expansion from the active state $\{0, w_1\}$, the arc labelled $y : \epsilon$ will benefit from a rescore threshold, while $x : \epsilon$ will not.

3.3. Optimization Techniques

There are four configurable parameters in this design. All four affect the search behavior with respect to keeping additional rescore-hypotheses.

The first parameter is the maximum number of words a $C \circ L$ state can reach and still be included in the rescorable state set. The $C \circ L$ is conceptually very much like a prefix tree; all words are reachable at the start and fewer words become reachable as more transitions are taken. The WFST composition of $C \circ L \circ G$ with weight pushing results in LM costs appearing on the arcs of the $C \circ L \circ G$ as early as possible while maintaining the same distribution. However, when a large number of words are reachable only the weight equal to the least-cost word can be pushed [8]. Hence, we consider keeping hypotheses only for $C \circ L$ states where the number of reachable words is below some threshold.

The second parameter is the rescore threshold, T^R. This value is the maximum score difference allowed between a hypothesis that meets our rescore condition and the best hypothesis. Too low and we cannot retain good hypotheses, too high and we are using search resources on useless hypotheses.

The third parameter is the maximum number of rescore hypotheses. If we have a large number of hypotheses allowed by the rescoring condition, we must reduce T^R as described earlier until the number of rescore hypotheses falls below this maximum limit again. This prevents the search space from growing too large.

Lastly, the fourth parameter controls the number of word-histories that are considered for a hypothesis when checking if it meets the rescore condition. An active search hypothesis may represent a number of distinct word histories. How those histories are maintained and how many are kept depends on the lattice implementation. We may choose to check if only the best word history of the hypothesis meets the rescore condition, any of them do, or some other approach.

We tuned these parameters using a simple grid search on a held-out set of development data.

4. Experiments

We performed experiments on two different contextual use-cases (strong context and weak context). We evaluated the proposed system in terms of speech recognition accuracy and in terms of computing resource usage.

In the first use-case we analyze system performance when strong contextual information is available. In this case, the user will say one of the phrases that is present in the contextual content. This use case corresponds to, for example, a particular dialog state in which a user is asked a question (e.g. "Do you want to send this email") for which there is a known set of answers (e.g. "yes", "no", ...) that are included in the contextual information. On this test set we measure the word error rate (WER) reduction, sentence accuracy (SACC) increase, and metrics which capture the effect of the rescore beam on the search space and CPU usage.

In the second use-case we analyze system performance when weak contextual information is available. In this case, the contextual information may or may not contain words or phrases that the user will say. This use case corresponds to, for example, biasing towards entities relevant in a particular geographic location (e.g. street names, restaurant names, ...), commands used on a particular device (e.g. device control commands), etc.

For this use-case we ran two experiments. One using geography-based weak contextual biasing, and one using only weak context appropriate for the device control commands.

We further investigated how much we would need to increase the beam size in a standard beam search algorithm to achieve the same accuracy gains achieved using the contextual search algorithm. In order to do that, we collected a set of utterances fixed by the contextual search and found the minimum beam size that is required for the standard beam search algorithm to correctly recognize those utterances. We compare the overall increase in search space from this approach to the contextual beam search.

In all of our experiments using rescore-aware pruning, we use a rescore threshold T^R approximately 30% higher than the standard beam, and a maximum rescore hypothesis count M^R approximately 2.5% of the standard maximum hypothesis count M. We only consider the best word history of a hypothesis when checking the rescore condition. Finally, the threshold on the maximum number of words reachable by a $C \circ L$ state was set to 1000.

4.1. Strong Context Experiments

The test set used for strong context experiments consisted of 4,000 utterances corresponding to voice search traffic. The data has been anonymized and does not contain personally identifiable information. For each of the utterances the contextual information contained the correct transcript. We apply biasing weights described in [3]. This use-case represents the best-case scenario for contextual biasing, the case in which contextual information contains exactly what the user is going to say.

Table 1: *WER and SACC observed on the strong context test set for three configurations.*

Case	WER [%]	SACC [%]
No context	15.0	67.42
Standard search with context	7.8	81.88
Contextual search with context	7.4	82.58

We experimented on this set using three different configurations. In the first experiment, we use no contextual information, relying only on the base LM. In the second experiment, contextual biasing is applied and the standard beam search used. In the third experiment, we use rescore-aware pruning along with the contextual rescoring LM. Using the contextual rescoring LM reduces the WER from 15.0% to 7.8% and increases the SACC from 67.42% to 81.88%. Using the contextual search further improves WER to 7.4% and SACC to 82.58%.

On this test set we observed a 2.8% increase in the number of states in the search space visited when the contextual search was used. Furthermore, on average, only 0.109% of states considered for expansion were allowed by the rescore beam. Comparing the CPU usage of the decoder between the standard search and the rescore-aware search shows a 4.8% increase, due to the slightly larger search space and the time spent checking the rescore condition.

4.2. Weak Context Experiments

In weak context experiments we use a randomly sampled slice of un-transcribed voice search traffic to evaluate our system. All data used in the experiments has been anonymized and does not contain personally identifiable information.

In the first experiment we apply weak (geographic) contextual information biasing on the slice of voice search traffic. For each utterance we fetch on-the-fly pre-compiled phrases that have been found to be salient for the query's geographical origin. This context is used to build a rescoring LM, using the approach described in our experiments in [3].

In the second experiment we use utterances containing media control commands. These commands can correspond to playing music, setting alarm or timer, controlling device volume, etc. Most common media control commands are included in an appropriate rescoring LM to be used for contextual ASR.

4.2.1. Experiment Setup

In order to evaluate the speech recognition accuracy of our proposed system on un-transcribed data we ran side by side (SxS) experiments. In SxS experiments two ASR systems are run in parallel: the baseline and the experiment system. The baseline system is a copy of our production service containing all our contextual biasing models. The experiment system is identical to the baseline except for the modified contextual beam search algorithm. Each SxS experiment is run until we collect 500 differences in recognition results between the baseline and the experiment system.

Each set of 500 differences is rated manually. A rater listens to the audio and then compares the transcript given from each ASR service. The rater is not told which transcript is which, nor that the experiment is evaluating a change for contextual search. Each transcript pair is rated as either an improvement (win), a regression (loss), or the same. An improvement need not indicate that the transcript from the new search side perfectly transcribes the audio, only that it does so more meaningfully than the baseline transcript. Similarly, two transcripts that are judged the same can be different in content but yield equally useful transcriptions.

An example of an improvement from our experiments is "the grease volume by 5%" becoming "decrease volume by 5%", where the audio was of low quality but recognizable to human ears as the latter. An example of a difference judged the same is "AutoZone 1st and Glenn" and "AutoZone first and Glenn". Each difference is graded by at least two raters, and a third is used if a tie-breaker is required.

4.2.2. Results

For each SxS experiment we analyze two metrics: the percentage of the test search queries that yielded a difference (Coverage), and the Win/Loss (W/L) ratio of the differences.

Table 2: *W/L ratio and percentage of traffic affected for search traffic and device control traffic.*

Source	Coverage [%]	Win/Loss
Search Traffic	0.179	2.6
Device Control	0.172	2.86

In both SxS experiments the proposed contextual search system outperforms the baseline system. Although a small percent of traffic is affected, the misrecognitions fixed that were observed were on difficult, long tail utterances. Those include utterances corresponding to accented speakers, utterances containing significant level of noise, or utterances with multiple speakers.

4.2.3. Comparison to Standard Search

We also investigated how much we would need to increase the beam size in a standard beam search algorithm to achieve the same accuracy gains achieved using the contextual search algorithm. We took 50 of the utterances that were found to be improved in this experiment and performed a grid search on the maximum active hypotheses M and standard beam threshold T to find the operating points that achieved parity with the rescore-aware pruning while keeping the smallest possible search space.

Table 3: *WER, SACC, and search space size of three analyzed configurations.*

Configuration	WER [%]	SACC [%]	Avg states searched
Baseline	63.8	0.0	355k
Rescore-aware	12.2	80	367k
Large beam	12.9	78	1066k

A beam size (T) increase of 21.4% and an approximately 100% increase in the maximum number of active hypotheses (M) was required to match the rescore-aware decoder performance. This led to a 200% increase in the size of the search space, compared to only 3.3% increase using rescore-aware pruning. It is clear from these results that the fact that rescore-aware pruning targets the search space growth only in directions of contextual relevance leads to optimal performance.

5. Conclusions and Future Work

We have described a system that fixes a class of search errors that arise in the decoding process of a contextual ASR system when contextual rescoring is performed on the lattice. We have demonstrated why those errors occur and how they can be fixed by an appropriate modification of the search algorithm to take advantage of the contextual information available at search-time. We analyzed the quality performance of this change on several test sets and use-cases showing clear accuracy improvements. We also analyzed the performance and cost of this algorithm in computing resources. The experimental results show that this technique is highly efficient and can be used on a general purpose ASR system to improve long-tail and hard-to-fix contextual misrecognitions.

In the future, we plan to use this work on class-based [7] and OOV search space extensions. We also plan to explore adaptively setting the increased beam size for contextual n-grams based on the base LM score of that n-gram, rather than choosing a fixed beam size for all contextual n-grams.

6. References

[1] P. S. Aleksic, M. Ghodsi, A. Michaely, C. Allauzen, K. B. Hall, B. Roark, D. Rybach, and P. J. Moreno, "Bringing contextual information to google speech recognition." in *INTERSPEECH*, 2015, pp. 468–472.

[2] A. Michaely, J. Scheiner, M. Ghodsi, P. Aleksic, and Z. Wu, "Unsupervised context learning for speech recognition," 2016.

[3] J. Scheiner, I. Williams, and P. Aleksic, “Voice search language model adaptation using contextual information,” in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 253–257.

[4] K. B. Hall, E. Cho, C. Allauzen, F. Beaufays, N. Coccaro, K. Nakajima, M. Riley, B. Roark, D. Rybach, and L. Zhang, “Composition-based on-the-fly rescoring for salient n-gram biasing,” in *Interspeech 2015*, 2015.

[5] G. Saon, D. Povey, and G. Zweig, “Anatomy of an extremely fast lvcsr decoder.” in *Interspeech*, 2005, pp. 549–552.

[6] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, “Openfst: A general and efficient weighted finite-state transducer library,” in *International Conference on Implementation and Application of Automata*. Springer, 2007, pp. 11–23.

[7] L. Vasserman, B. Haynor, and P. Aleksic, “Contextual language model adaptation using dynamic classes,” in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 441–446.

[8] D. Rybach, R. Schlüter, and H. Ney, “A comparative analysis of dynamic network decoding,” in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5184–5187.