Bringing Contextual Information to Google Speech Recognition

Petar Aleksic, Mohammadreza Ghodsi, Assaf Michaely, Cyril Allauzen, Keith Hall, Brian Roark, David Rybach, Pedro Moreno

Google Inc.

{apetar,ghodsi,amichaely,allauzen,kbhall,roark,rybach,pedro}@google.com

Abstract

In automatic speech recognition on mobile devices, very often what a user says strongly depends on the particular context he or she is in. The n-grams relevant to the context are often not known in advance. The context can depend on, for example, particular dialog state, options presented to the user, conversation topic, location, etc. Speech recognition of sentences that include these n-grams can be challenging, as they are often not well represented in a language model (LM) or even include out-of-vocabulary (OOV) words.

In this paper, we propose a solution for using contextual information to improve speech recognition accuracy. We utilize an on-the-fly rescoring mechanism to adjust the LM weights of a small set of n-grams relevant to the particular context during speech decoding.

Our solution handles out of vocabulary words. It also addresses efficient combination of multiple sources of context and it even allows biasing class based language models. We show significant speech recognition accuracy improvements on several datasets, using various types of contexts, without negatively impacting the overall system. The improvements are obtained in both offline and live experiments.

1. Introduction

The impact of quality of speech recognition on user experience on mobile devices has been significantly increasing with increase in voice input usage. Voice input is used to perform search by voice, give specific voice commands, or ask general questions. The users expect their phones to keep on getting smarter and to take into account various signals that would improve the quality of communication with the device and overall user experience.

In this effort, utilizing contextual information plays a great role. The context can be defined in a number of ways. It can depend on the location that the user is in, on the time of the day, the user's search history, the particular dialog state that the user is in, the conversation topic, the content on the screen that the user is looking at, etc. Very often the amount of information about the context can be very small, consisting of only a few words or sentences. However, if the context is relevant it can significantly improve the speech recognition accuracy, if consumed appropriately by the speech recogntion system.

In this paper we present a system that uses contextual information to improve speech recognition accuracy. Our solution works well for both large contexts and contexts consisting of only several words or phrases. We use a framework for biasing language models (LM) using n-grams as the biasing context [1]. The n-grams and corresponding weights, calculated based on the reliability of the context, are represented as a weighted finite-state transducer [2, 3], contextual model. We introduce several approaches for creating contextual models from the context, as well as methods for combining the score from the main language model and the contextual model. In addition, we address the issue of handling out-of-vocabulary (OOV) words present in the provided context, by using a class specific language model, as described in section 2.

One can view this approach as a generalization of cache models [4, 5, 6], which have been used to personalize language models based on recent language produced by the individual whose utterance is being recognized. Our approach derives the biasing n-grams from varied sources beyond an individual's prior utterances and makes use of more complex methods for mixing with the baseline model than the fixed interpolation or decay functions typically used with recency cache models [4, 5]. See also the discussion of related work in [1].

We organize the paper as follows. In section 2 we present the approach we used to perform on-the-fly n-gram biasing of the language model towards context present in the contextual model. In section 3 we present various approaches for creating a contextual model from the provided context. Finally, in section 4, we describe the test sets used in our experiments and present all of our experimental results.

2. Contextual language model biasing

In this section we describe the language model biasing framework we use, and how it handles class-based language models and OOVs.

2.1. General approach

We used the framework for biasing language models using n-grams, introduced in [1]. In this framework, a small set of n-grams is compactly represented as a weighted finite-state transducer [7]. An on-the-fly rescoring algorithm allows biasing the recognition towards these n-grams. The cost from the main language model G is combined with the cost from the contextual model B as follows:

$$s(w|H) = \begin{cases} s_G(w|H) & \text{if } (w|H) \notin \mathcal{B} \\ C(s_G(w|H), s_B(w|H)) & \text{if } (w|H) \in \mathcal{B} \end{cases}, \quad (1)$$

where $s_G(w|H)$ is the raw score from the main model G for the word w leaving history state H and $s_B(w|H)$ is the raw score for the biasing model. Observe that this approach only modifies the LM scores of n-grams, Hw, for which the biasing model provides an explicit score. This differs from regular language model interpolation and is motivated by the fact that the support of the biasing model is much sparser than that of the main language model.

[1] offers the following alternatives for the operation C

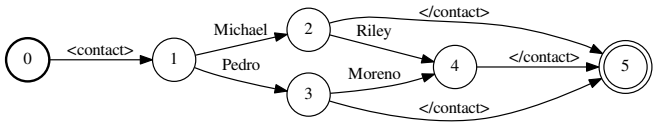

Figure 1: Example of a class grammar with decorators for the "$CONTACTS" class.

used to combine the scores. The first approach corresponds to using log-linear interpolation:

$$C'(s_G(w|H), s_B(w|H)) = \alpha * s_G(w|H) + \beta * s_B(w|H). \quad (2)$$

Since our costs are negative-log conditional probabilities, this simply corresponds to linear interpolation in the log-domain.

Finally, [1] also provides a mechanism that restricts the biasing to be applied only if it reduces the cost. In equation 3 we define the *positive biasing* function which applies this restriction:

$$C(s_G(w|H), s_B(w|H)) = \min(s_G(w|H), C'(s_G(w|H), s_B(w|H))). \quad (3)$$

Dynamic decoding of input speech is performed similarly to what is described in [8]. Specifically, given a vocabulary V we generate a lattice from the alphabet $\Sigma = V \cup \{\epsilon\}$. Given a CLG (a composition of the context-dependent phone model, lexicon, and main language model), we perform time-synchronous decoding via beam search. As in [8], a pseudo-deterministic word-lattice is built during decoding. It is at this point where we apply the on-the-fly rescoring [9] with the contextual biasing model as described in [1].

2.2. Biasing class-based language models

Our main language model is class-based [10, 11, 12]. Examples of classes are address numbers, street names, dates, and contact names. The last being an example of an utterance-dependent user-specific class.

We might want to bias towards the whole class in some context. For instance, we might want to bias towards "call $CONTACTS" or "directions to $ADDRESSNUM $STREETNAME" instead of being limited to simply biasing towards some instantiations of the classes (e.g. "call Michael" or "directions to 111 Eight Avenue").

Our language model consists of: (a) a top-level n-gram language model over regular words and class labels and (b) for each class c, a class grammar G_c over regular words that might be utterance-dependent. All components are represented as weighted automata. At run-time, this model is expanded on-demand into a weighted automata G using the replacement operation as described in [10]. In this approach, class-based biasing is achieved by: (1) Modifying each class grammar to insert decorators allowing us to keep track of whether words in the hypothesis word lattice were generated by the top level LM or by one of the class grammars. This corresponds to using $G'_c = <c> \; G_c \; </c>$ as class grammar for class c where $(<c>, </c>)$ is the decorator pair for c (see Figure 1). (2) Allowing n-grams containing class labels in the contextual biasing model. (3) Treating decorator-delimited phrases as their corresponding class-label during rescoring. This is achieved by composition on-the-fly the word lattice with the transducer T described Figure 2 and then applying the biasing model as described in the previous section.

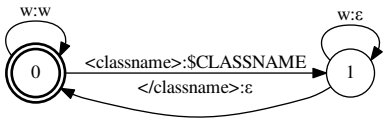

Figure 2: Transducer T maps decorator-delimited phrases back to the corresponding class label.

2.3. Handling out-of-vocabulary words

The contextual model might contain words that do not appear in the base vocabulary of our base language model. We want to be able to add these OOV words to our base LM at the unigram state so that they can be hypothesized and rescored accordingly by the contextual model.

We achieve this by leveraging class-based language modeling approach. We introduce a "$UNKNOWN" class that only appears at the unigram state of the LM. At recognition time, we extract the set of OOV words from the contextual model for the considered utterance. We then create a "$UNKNOWN" class grammar representing these words, as a monophone-to-word transducer as described in [10]. In this instance, we do not add decorators to the class grammars, since we want to rescore the individual OOV words in the biased contexts and not the "$UNKNOWN" class.

3. Constructing the contextual model

The context we use for biasing can consist of hundreds of phrases or only a handful of phrases. Each phrase is a sequence of one or more words. For example, the following phrases may be used as the context for an utterance: "Hotels in Manhattan", "Holiday Inn", "Cheap flights to New York City".

When biasing, we want to allow partial matching to the context. For example, given the context above, we might also want to bias towards "Cheap hotels in New York".

In general, if the size of the context is large enough that a regular language model can be constructed from it, then one can use the LM costs as biasing scores. (In that case, the interpolation would be a standard interpolation between two LM costs.) However, often the context available is too small for using this approach. We developed methods that address this case.

In this section, we discuss how we select biasing n-grams and their scores, given a set of context phrases such as above.

3.1. Extracting and scoring n-grams

We want to bias more heavily towards higher order (longer) n-grams. This is because of two related reasons: The first is that we want to reward longer exact matches between the context and the recognition result. The second is that biasing towards shorter n-grams has a larger negative effect on the recognition of general (out of context) queries.

One simple scoring function that satisfies the above requirements is the length-linear function, where n is the length of Hw. That is:

$$s_B(w|H) = f_1(\text{length}(Hw)) = (n-1)p_2 + p_1 \quad (4)$$

where p_1 and p_2 are parameters that control the strength of biasing, their values depending on the quality of the context. These parameters can be learned on a transcribed development data set with context.

The length-linear function would assign the same score to all n-grams of the same order. However, because the final cost used by the recognizer is an interpolation of the biasing score and the original LM cost, the effect of the biasing score depends on the interpolation function.

Since we want to bias more heavily towards longer n-grams, we would want $s_B(n)$ to be a decreasing function of n, i.e. $p_2 < 0$.

The main limitation of the length-linear function is that the cost of various n-gram orders are interdependent. A slightly more general function would assign independent scores to each of the n-gram orders. In our system, we observed diminishing gains beyond specifying scores for unigrams and bigrams only. (Note that, similar to back-offs mechanism in LMs, the biasing model will use the score of the lower order n-gram if the longer one is absent.)

We define the unigram-and-bigram function as:

$$s_B(w|H) = f_2(\text{length}(Hw)) = \begin{cases} p_1 & : n = 1 \\ p_2 & : n \geq 2 \end{cases} \quad (5)$$

The unigram-and-bigram function is more robust and easier to interpret, compared to the length-linear function. We therefore used the unigram-and-bigram function in most of the experiments presented in this paper.

3.2. Sentence boundaries

As mentioned, biasing towards unigrams can be detrimental to the general query recognition performance. But what if some or all of the context phrases contain only one word? For example, in one of our test sets (confirmation) the context consists of the phrases "yes", "no", and "cancel". If we were to bias towards these unigrams heavily, we may get recognition results that contain repetitions of these words, such as "no no no . . . ".

We can avoid this outcome by appending sentence boundary tokens ("<S>" and "</S>" in our case) to each phrase in the context, before extracting the biasing n-grams. Then, in the above example, we would bias towards bigrams such as "<S> no" and "no </S>" much more than we bias towards the unigrams.

4. Experimental results

In this section we describe our test sets, experimental setup, and analyze the experimental results. All test sets used have been anonymized.

4.1. Corpora

The experiments described below use various test sets in American English. All of the test sets were manually transcribed. The context for some test sets is defined per utterance (e.g. test set "Entities and location"), whereas for others the context is constant for the whole test set (e.g. test set "Confirmation"). Several experimenal setups were used to evaluate the positive effect of relevant context and the negative effect (overtriggering) of irrelevant context. In the baseline setup, experiments are run with no context provided. In order to evaluate the positive effect of relevant context we use the following setups: (1) In sets with per-utterance context, we attach to each utterance its relevant context. (2) In sets with fixed context, the same context is attached to every utterance

In order to evaluate the negative effect of relevant context we use the following setups: (1) In sets with per-utterance context, we attach to each utterance 100 irrelevant contexts randomly selected from other utterances. We call this a negative set. (2) In sets with fixed context, we attach the fixed context to a set of utterances for which the context is irrelevant. We call this an anti-set.

4.1.1. Entities and location

This test set contains 876 utterances. Each utterance contains the name of an entity and/or the name of a location e.g. "Directions to Sky Song in Phoenix, Arizona". The context is defined per utterance, and is a list of locations and entities, e.g. {"Sky Song", "Phoenix, Arizona"}.
Test set variants: **entities_pos, entities_neg, entities_baseline**

4.1.2. Confirmation

This test set contains 1000 utterances. All queries correspond to a state where the user is provided with the choice to confirm or cancel some action. The context is the same for all utterance, and it consists of the words {"yes", "no", "cancel"}.
Test set variants: **ync_pos, ync_baseline and anti_ync**. anti_ync is an anti-set consisting of 22k utterances not related to confirmation/cancellation states.

4.1.3. Hard n-grams

This testset consists of 2,704 utterances. All utterances in this testset contain n-grams with high LM costs, for $n \in [2, 7]$. The context, defined per utterance, is a list of high-cost n-grams.
Test set variants: **costly_pos, costly_neg, costly_baseline.**

4.1.4. Class based (numeric)

This testset contains 816 utterances, each containing some type of number in the transcript, e.g. "Set alarm for 5:30 p.m. today". The context is defined per utterance and consists of a list of the transcripts with class members replaced by their class symbol (e.g. "Set alarm for $TIME p.m. today"). The context for each utterance contains the utterance's modified transcript.
Test set variants: **numeric, numeric_baseline**

4.1.5. Class based (contacts)

This testset contains 10670 utterances. All utterances correspond to contact calling voice commands, e.g. "Call James Brown". Similar to numeric testset the context is created by name class members being replaces by "$CONTACTS" in transcripts.

4.2. Recognition accuracy with biasing

We measured the effect of biasing on our test sets using both of the functions introduced in section 3.1. We then measured the effects of each of the features that our biasing implementation supports. Finally, we show how we can control the strength of the biasing by varying a range of parameters of our biasing score function.

Table 1 shows the effect of biasing versus the baseline (i.e. no biasing). "bias 1" uses the length-linear scoring function, and "bias 2" uses the unigram-and-bigram function. Both biasing tests use the positive biasing interpolation function in equation (3), however "bias 1" uses $(\alpha, \beta) = (0.25, 1)$ whereas "bias 2" uses $(\alpha, \beta) = (0, 1)$, which is effectively the same as using $\min(s_G(w|H), s_B(w|H))$ for interpolation. The values of α and β control the interpolation of main LM costs and biasing scores based on equation (3).

Test set	Baseline	bias 1	bias 2
entities_pos	8.9	7.2	7.2
entities_neg	8.9	9.0	9.0
ync_pos	18.8	10.4	11.0
anti_ync	10.9	10.9	10.9
costly_pos	12.9	4.2	6.1
costly_neg	12.9	13.8	13.8
numeric	11.0	4.7	5.7
contacts	15.0	2.8	3.2

Table 1: WER(%) for baseline vs two biasing methods. bias 1: length-linear, $(\alpha, \beta) = (0.25, 1)$ and $(p_1, p_2) = (0, -0.4)$. bias 2: unigram-and-bigram, $(\alpha, \beta) = (0, 1)$ and $(p_1, p_2) = (7, 3)$.

Test set	bias 2	bias 2.a	bias 2.b	bias 2.c
entities_pos	7.2	7.2	7.3	7.4
entities_neg	9.0	8.9	9.0	9.0
ync_pos	11.0	15.0	11.6	11.0
anti_ync	10.9	10.9	10.9	10.9
costly_pos	6.1	6.5	6.7	6.1
costly_neg	13.8	13.6	13.8	13.8
numeric	5.7	6.1	6.0	5.9
contacts	3.2	5.1	3.2	3.2

Table 2: The effect of biasing features on WER(%): bias 2: With all features, same as in Table 1. bias 2.a: Without sentence boundaries. bias 2.b: Without case variants. bias 2.c: Without OOV support.

Table 2 compares the effect of having each of the following features *disabled*:

bias 2.a. Add sentence boundaries to the context.

bias 2.b. Include upper/lower case variants of the context.

bias 2.c. Support OOV words in the context.

Disabling sentence boundaries has a particularly detrimental effect on our ync_pos test set. It also negatively affects contacts, numeric and costly_pos test sets. As mentioned in section 3.2, the reason is that in the ync_pos test set, the context and most of the expected transcripts are unigrams. Adding sentence boundaries allows us to bias these contexts at the bigram level, which can be safely biased more heavily. In contrast, trying to achieve the same result by increasing unigrams bias would result in a sharp increase in insertion errors.

Disabling case variants has a negative effect on the recognition resuls for the test sets for which the context includes the case variant less probable in the LM. The worst effect is on the costly_pos and numeric test sets. This feature does not have a noticeable effect on the negative test sets, so it can be safely turned on by default. The third feature, support for OOV words in the context, reduces WER on test sets that include OOVs and has no effect otherwise.

Finally, we present WER for our entities and location test sets at a range of operating points (set of values for parameters in the scoring function). This test set contains context phrases of various lengths, case variants, and OOVs. In Table 3 the WER for the positive and negative tests are shown side by side, for various pairs of values of (p_1, p_2) in equation (5). The point $(0, 0)$ corresponds to baseline WERs as the context is ignored.

At the lowest level of biasing, $(p_1, p_2) = (10, 5)$, the negative test is not affected (the WER is equal to baseline). However the positive WER is already better than the baseline. As

p_1	p_2							
	-1		0		1		5	
-2	30.8	*43.3*	30.5	*42.8*	32.7	*42.0*	35.9	*42.1*
0	7.2	*9.3*	**8.9**	***8.9***	7.3	*9.0*	7.7	*8.9*
6	6.6	*9.6*	7.3	*9.1*	**6.6**	***9.2***	7.3	*9.0*
10	6.7	*9.4*	8.2	*8.9*	7.0	*9.0*	7.5	*8.9*

Table 3: WER(%) for entities_pos (using regular font) and entities_neg (using *italics*) over a range of (p_1, p_2) values for the unigram-and-bigram scoring function (equation (5)).

the strength of bias is increased, the WER for the negative test increases monotonically, but the WER for the positive test set decreases, up to a certain minimum, after which it also starts increasing. This is because the context starts to cause errors in the parts of the utterance that are not supposed to be biased. At the extremely high biasing level of (-2, -1), both the positive and negative tests are significantly worse than baseline.

The operating point of $(7, 3)$ used in Table 1 and Table 2 is a relatively conservative operating point, which has minimal effect on the negative test. This operating point was chosen to balance positive and negative performance on several different test sets. For the test set used in Table 3, a more aggressive operating point of $(6, 1)$ results in WERs 6.6% and 9.2%, respectively, on the positive and negative tests (baseline is 8.9%).

4.3. Live biasing experiments

In order to further validate that our system improvements are beneficial we ran a live experiment. In our experiment, a percentage of the production traffic is cloned and sent to two speech recognition systems. We focused only on the traffic corresponding to the confirmation dialog state, that is, the state in which a user is asked to respond with one of the words “yes”, “no”, “cancel”. The first system was used as the baseline while the second used the biasing methodology described in this paper. In the biasing system, for each of the utterances we used the fixed biasing context consisting of three words described above.

During our experiment approximately 30,000 utterances were processed by each system. This was done anonymously and on-the-fly. We compared the performance of the two systems by using sentence accuracy as metric. Using the biasing methodology and optimal operating point described in section 4.2 resulted in a sentence accuracy increase of 8% relative. This was significant with $p < 0.1$.

5. Conclusion

In this paper, we describe an approach for biasing speech recognition towards provided contextual information. We analyze various types of context, describe context preprocessing techniques, and provide a solution to OOVs present in the context. We also present biasing functions used to adjust LM scores based on provided context. We conducted experiements using several datasets with various types of contextual information provided. The results obtained show that the proposed methodology can significantly improve speech recognition accuracy when reliable contextual information is available. For example, on our confirmation (ync) testset, WER relative reduction of 44% is achieved on positive (ync_pos) testset without any WER changes on the negative test set (anti_nyc). Furthermore, we show that speech recognition gains are achieved withouth causing overtriggering on queries not related to the context.

6. References

[1] 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.

[2] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *CIAA 2007*, ser. LNCS, vol. 4783, 2007, pp. 11–23, `http://www.openfst.org`.

[3] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," in *Handbook of Speech Processing*, Y. H. Jacob Benesty, Mohan Sondhi, Ed. Springer, 2008, pp. 559–582.

[4] R. Kuhn and R. De Mori, "A cache-based natural language model for speech recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 6, pp. 570–583, 1990.

[5] P. R. Clarkson and A. J. Robinson, "Language model adaptation using mixtures and an exponentially decaying cache," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE Internation al Conference on*, vol. 2. IEEE, 1997, pp. 799–802.

[6] S. Besling and H.-G. Meier, "Language model speaker adaptation," in *Fourth European Conference on Speech Communication and Technology*, 1995.

[7] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, pp. 69–88, 2002.

[8] G. Saon, D. Povey, and G. Zweig, "Anatomy of an extremely fast LVCSR decoder," in *in Proc. Interspeech*, 2005, pp. 549–552.

[9] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 4, pp. 1352–1365, 2007.

[10] P. Aleksic, C. Allauzen, D. Elson, A. K. D. M. Casado, and P. J. Moreno, "Improved recognition of contact names in voice commands," in *ICASSP 2015*, 2015.

[11] L. Vasserman, V. Schogol, and K. Hall, "Sequence-based class tagging for robust transcription in ASR," in *Submitted to Interspeech*, 2015.

[12] P. F. Brown, V. J. D. Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.