Composition-based on-the-fly rescoring for salient n-gram biasing

Keith Hall, Eunjoon Cho, Cyril Allauzen, Francoise Beaufays, Noah Coccaro, Kaisuke Nakajima, Michael Riley, Brian Roark, David Rybach, Linda Zhang

Google, Inc.

{kbhall,ejcho,allauzen,fsb,noahc,kaisuke,riley,roark,rybach,lindazhang}@google.com

Abstract

We introduce a technique for dynamically applying contextually-derived language models to a state-of-the-art speech recognition system. These generally small-footprint models can be seen as a generalization of cache-based models [1], whereby contextually salient n-grams are derived from relevant sources (not just user generated language) to produce a model intended for combination with the baseline language model. The derived models are applied during first-pass decoding as a form of on-the-fly composition between the decoder search graph and the set of weighted contextual n-grams. We present a construction algorithm which takes a trie representing the contextual n-grams and produces a weighted finite state automaton which is more compact than a standard n-gram machine. Finally, we present a set of empirical results on the recognition of spoken search queries where a contextual model encoding recent trending queries is applied using the proposed technique.

Index Terms: speech recognition, language modeling

1. Introduction

The context within which an utterance is made can influence the expectation of what was spoken. For example, the location of an individual when making a voice query on their mobile phone can influence the likelihood of certain words and phrases, such as place names in the same region as the speaker. Other contextual side information could also be of similar utility, including temporal or seasonal information, previously observed individual language use or actions, or the specific application being used with spoken input. In the case of applications such as voice search, particularly newsworthy or trending items may have higher likelihood in the moment than estimated by the static, general-purpose language model.

In many such cases, the relevant information is not static, and hence must be derived and served on-the-fly. The amount of data associated with any particular source of side information may not be of particularly large scale and often is not really intended to be a representative sample over all strings of words from the vocabulary. For example, one may have access to a list of frequently queried locations within a certain distance from the speaker; or a relatively small but recent sample of the speaker's previous queries, such as is often used to produce a recency cache [1]. Models derived from such sources are generally not robust, large vocabulary models that would be used on their own for general speech recognition. Rather, they are intended to be used in conjunction with a baseline large vocabulary model, to bias recognition towards words and phrases for which there is external evidence of increased (or possibly decreased) likelihood.

Most approaches to modifying a static, baseline language model with contextual or in-domain information takes the form of building an in-domain model and performing model interpolation, i.e., mixing the probability distributions in one of a number of standard ways [2]. In the current paper, we explore a new method for exploiting in-domain or contextually salient n-grams, which biases the weights of a subset of n-grams while leaving the weights of all others alone. This is effected by composing a weighted finite-state transducer (WFST) representing the baseline model with a compact WFST representation of the set of n-grams being biased. This compact representation uses special arcs (ϵ and ρ arcs, see details in section 2.1) so that every n-gram in the original model survives the composition with a weight that is only changed from the original weight for the n-grams in the specified subset. This contrasts with typical model mixing or rescoring methods, where the scores of all n-grams are modified. One might think of these as generalizations of cache language models [1], whereby collections of *salient* n-grams (however that saliency is determined) are used to bias recognition towards them. The approach in this paper generalizes the sources of such small n-gram collections beyond user generated language, as well as the kinds of composition functions that are used to combine the scores with the baseline model. See Section 4 for further discussion of related work.

We present results using biasing WFSTs derived from recently trending n-grams within a voice search application (for additional applications see [11]). The models are applied in a rapid rescoring scenario, meaning that the model is composed with a very large set of possible extensions to the lattice being constructed prior to pruning during first-pass recognition. We demonstrate significant improvements in recognition accuracy by the use of these methods on a test set of expected fast rising queries, while remaining neutral on general voice search traffic.

2. Methods

To perform composition-based on-the-fly rescoring, a set of n-grams and target costs for those n-grams must be selected, and this weighted set must be compiled into a data structure for use. Before discussing (in section 2.2) how the n-grams were selected in the case studied in this paper, we first present methods for compiling the selected set into an efficient finite-state representation and combining with the decoder graph.

2.1. Composition-based on-the-fly rescoring

2.1.1. Weighted finite-state compilation

We compile a given set of n-grams into a weighted finite-state transducer (WFST) representation. Figure 1 presents a standard trie structure representing a small set of n-grams. Each node in the trie has a suffix link (represented with a dotted line arc) to

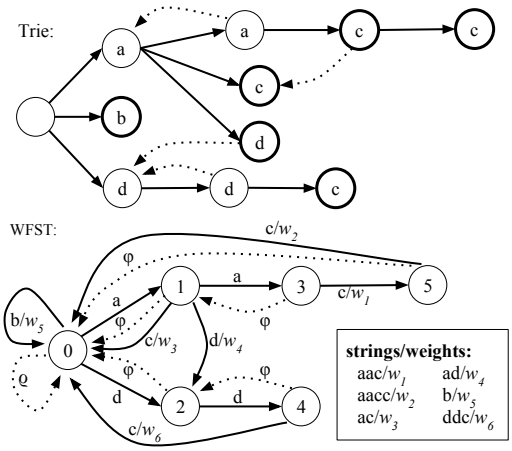

Figure 1: Example of trie representation of n-grams in a given set, and compact (minimal) WFST representing the same set. Bold circles in the trie represent leaf nodes, and dotted arcs represent suffix links. In the WFST, dotted arcs represent either ϕ or ρ transitions, which are traversed only in the absence of a matching symbol.

the node in the trie representing the longest proper suffix of the node, or to the root node if no proper suffix exists in the trie. (Suffix links to the root are omitted for clarity.) Bold circles represent leaf nodes, and the path from root to leaf determines the n-gram being stored.

The WFST below the trie in Figure 1 encodes the same set of n-grams (plus some extra information) more compactly. State 0 represents the initial state, and there is a state for every proper prefix of an n-gram in the set: state 1 represents the prefix 'a'; state 2 represents 'd'; and states 3, 4 and 5 represent 'aa', 'dd' and 'aac', respectively. Each of these states is reached from the initial state via a path of ascending arcs labeled with the prefix n-gram. Each state has a failure arc (denoted with a dotted line and labeled with either ϕ or ρ) that is traversed only when trying to match with a symbol that does not label any arc leaving that state (an 'otherwise' arc). A ϕ arc is traversed without consuming the symbol being matched, just as with an ϵ arc. The ρ arc leaving state 0 consumes the symbol being matched (leaving it unchanged), so that the WFST successfully composes with any arbitrary input sequence. For example, the sequence "a d g b" would successfully compose with the WFST in Figure 1 by following sequence: start in state 0; 'a' transition from state 0 to state 1; 'd' transition to state 2; 'ϕ transition to state 0 (because there is no arc with 'g' leaving state 2); ρ transition to state 0 (no arc with 'g' leaving state 0); and finally 'b' transition to state 0.

Note that there are two kinds of transitions in the WFST – those which carry the weights that were provided with the list of n-grams, and those that have no labeled weight, but are required in order to access the correct states for detecting when an n-gram in the set is found. See section 2.1.2 for details on how the generalized composition works with weight-carrying arcs as well as non-weight-carrying arcs.

This WFST structure is similar to standard WFST n-gram representations [3], but has some key differences. Whereas in standard n-gram WFSTs, failure transitions are used to encode backoff arcs that have as destination the backoff history (the suffix of length exactly one symbol less than the current history), in the current representation the failure arcs point to the longest suffix that has a state in the WFST. Hence state 5 in the WFST in Figure 1 (representing the history 'aac') has a failure arc pointing to state 0 (no history) because there is no proper suffix of 'aac' that has a state in the WFST. In other words, there is no suffix of 'aac' that is a proper prefix of an n-gram in the set. Second, unlike a backoff n-gram model, the failure arcs do not carry a backoff weight, but are non-weight-carrying arcs. Finally, this WFST must distinguish in general between weight-carrying and non-weight-carrying arcs, the latter including failure arcs and those required to reach weight carrying arcs from the initial state.

We note that this WFST encoding is very similar to that for log-linear models in [4], and that paper proved that this representation is minimal for a given set of n-gram features.

2.1.2. Generalized Composition

We introduce the notion of a generalized composition[1] that we will apply between the n-gram WFST and the general language model being used in the first-pass decoder. Standard composition under a tropical semiring combines the scores of matched arcs (arrived at via matched paths), by summing the scores. In the context of probabilistic n-gram language models, we represent the score of a word given it's history as the negative log conditional probability, $-\log(p(w|H))$. Composition of two n-gram models in the tropical semiring is equivalent to multiplying the probabilities of each model[2].

Given the compact biasing WFSTs generated from our contextual source, we adopt a general mechanism for combining model scores, defined in equation 1.

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

Here, $s_G(w|H)$ is the raw score from the baseline grammar for the word w leaving history state H and $s_B(w|H)$ is the raw score for the biasing model, if such a score is provided. We define the function $C(s_G(w|H), s_B(w|H))$ to replace the semiring multiplication operation, e.g., addition in the tropical semiring. Some alternative score combining functions follow.

$$C'(s_G(w|H), s_B(w|H)) = \quad (2)$$
$$\begin{cases} \alpha * s_G(w|H) + \beta * s_B(w|H) & \textit{// LL} \\ -\log(\alpha * \exp(-s_G(w|H)) + \beta * \exp(-s_B(w|H)) & \textit{// LIN} \end{cases}$$

Equation 2 defines the function C' which combines the general language model score with the biasing model in either a log-linear (labeled *LL*) or linear (labeled *LIN*) manner. As our scores are negative log conditional probabilities, a sum of the scores is equivalent to a log-linear combination of the probabilities. Finally, we provide a mechanism that restricts the biasing to only be applied if it improves the score. In equation 3 we define the *positive biasing* function which applies this restriction.

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

2.1.3. Dynamic Decoding

We perform dynamic decoding of input speech similar to that described in [5]. Given a CLG (a composition of the context-dependent phone model, lexicon, and general language model), we perform time-synchronous decoding via beam search. As in

[1] This generalization of composition may no longer be compatible with the semiring to give invariance with respect to the topology of its inputs. We retained the name, qualified by *generalized*, since we do not need that invariance in our applications and the underlying algorithmic steps are otherwise identical.

[2] We omit the details of standard n-gram model merging as it requires proper merging of backoff transitions, etc.

[5] we build a pseudo-deterministic word-lattice during decoding. It is at this point where we apply an on-the-fly composition [6, 7] with the above WFST representing the set of n-grams to bias. In order to perform this during decoding, we build the cross-product of the decoder states and the biasing WFST states. When a word-emitting arc is traversed, we look up the extension of the previous state in the biasing WFST, and if found, we combine this state with the current decoder state.

2.2. Training contextual models

We explore a specific contextual model that is dependent on time, which we refer to as a recency model. It is trained from queries that appear in general search traffic more often within a specific recent time period than in the past. We use this model to bias the baseline model until it is replaced with a new recency model. Although there are no inherent constraints on how frequently we can update the recency model, for this paper they are updated daily. A pipeline extracts fast rising queries from our search traffic and uses them to train the recency model.

Given a set of fast rising queries, it can be tricky to assign the target scores $s_B(w|H)$ required for the sequence of words in a selected query. Should all substrings of the selected queries be biased, or only full queries? How can the query counts derived from fast rising query selection be compared to the probabilities in the baseline model in order to derive the scale of the bias, particularly since the selected data set is very small relative to the general language model training data?

For this paper, we build a smoothed trigram model[3] from the selected queries, then score every prefix of queries in the collected set (including full queries) with the language model score for the word given the history. For example, given the query *'storm in new york'*, we select the n-grams *'storm'*, *'storm in'*, *'storm in new'*, *and 'storm in new york'* to appear in the set of biased n-grams, with the corresponding weights derived from the model: -log $p(storm|<s>)$, -log $p(in|<s>, storm)$ and so on. All higher order n-gram probabilities are approximated with the smoothed trigram probabilities, but the full higher order n-gram is included in the set of selected n-grams. Biasing only prefixes of queries, rather than all substrings (as an n-gram model would do), has the benefit of compactness (see section 2.1.1), but also targets the bias more strictly on the queries of interest, e.g., we can bias *'storm in new york'*, without having to worry about the model biasing the suffixes *'in new york'*, *'new york'*, etc., in all contexts.

3. Experiments

We use two manually transcribed test sets to evaluate the performance of biasing WFSTs in the context of recency, drawn from randomized, anonymized voice-search traffic. The first test set VOICE-SEARCH (41295 words) is a sample from general voice search traffic, and tracks any regressions that appear by biasing recent queries. The other test set RECENCY (25254 words) contains utterances we expect are fast rising queries. We mark an utterance as a fast rising query if the pronunciation sequence of that utterance is high for that day compared to the average over the past 5 weeks. The test sets have utterances over the course of 3 months (201402 - 201404), and we have a recency model pre-built for each day. For each utterance we pick the most recently built recency model, and use it to bias the baseline model.

[3] A low order model is used due to the relative sparsity of data.

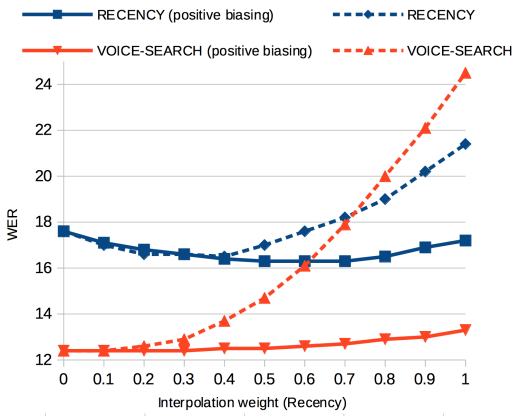

Figure 2: WER with varying interpolation weights for recency, α. Interpolation weight for the baseline model is $1 - \alpha$.

3.1. Model interpolation

We explore the effect of different interpolation weights with log-linear interpolation between the baseline model and the recency model. With *positive biasing* (Eq. 3), Figure 2 shows that we can get more than 7% relative WER gain when mixing the two models with equal interpolation weights, with a relatively small loss on general voice search traffic. As expected, there is a pair of interpolation weights that give us maximum WER gain on the RECENCY test sets, while the WER for VOICE-SEARCH increases with heavier biasing towards the recency model. Figure 2 also shows the importance of positive biasing in the recency model. Without positive biasing, the WER increases to 17.0 for RECENCY and 14.7 for VOICE-SEARCH with equal interpolation weights.

In practice, we can find the optimal operating pair of interpolation weights on a held out dev set, and use those instead. A simple 2-fold cross validation shows that, $\alpha = 0.7$, results in maximum WER gain on the RECENCY dev set, and with that pair of interpolation weights we get a similar relative WER gain of 7% on the held out RECENCY test set.

3.2. Biasing WFST vs N-gram model

In order to compare the effect of the biasing WFST representation with the standard n-gram model, we use both representations to do on-the-fly rescoring for recency models. With equal interpolation weights of 0.5, on-the-fly rescoring with a standard n-gram model results in 16.5 WER on the RECENCY test set and 12.6 on the VOICE-SEARCH test set. This is close to the WER with the biasing WFST representation, which achieves 16.2 and 12.5, respectively. Although the n-gram model does properly handle backoffs, it biases all possible substrings of extracted queries (as mentioned with the examples in Section 2.2), whereas the biasing WFST selectively biases only prefixes of extracted queries.

The use of a biasing WFST also enables the decoder to generate a better lattice for second pass rescoring, as relevant words will be biased towards being included in the final lattice. This can be measured using oracle normalized sentence accuracy (ONSACC) which calculates the oracle sentence accuracy from the generated lattice. The biasing WFST increases ONSACC from 77.7 to 78.9 for the RECENCY test set over the baseline, and from 84.8 to 84.9 for VOICE-SEARCH test set.

Yet another benefit of the biasing WFST representation is its compact size. As shown in Table 1, the biasing WFSTs have on average an approximately 60% reduction in bytes and num-

Model	Size(MB)	States	Arcs
N-gram	2.69	46579	143905
Biasing WFST	1.08	16436	50296

Table 1: Average size comparison of N-gram and biasing WFST representations.

ber of states and arcs versus the n-gram representation.

3.3. On-the-fly rescoring vs model mixed language model

We also compared the difference between applying a biasing WFST on-the-fly versus simply applying a similar composition off-line and using the composed model as the first-pass LM. Off-line composition is effectively a form of model merging, where we derive a model which produces the scores that would have been obtained by combining the scores from each model (the baseline and the biasing model). We do this first by transforming the biasing WFST to a full n-gram model topology and then mapping both models to a topology that contains all n-gram states needed by the pair of models. Finally, we apply our generalized composition function to combine the scores.

	RECENCY		VOICE-SEARCH	
Model	1st-pass	2nd-pass	1st-pass	2nd-pass
Baseline	20.1	17.6	13.4	12.4
On-the-fly	17.7	16.2	13.5	12.5
Off-line	17.3	15.8	13.5	12.4

Table 2: Results, reported in WER, comparing the effect of no biasing model (Baseline), the on-the-fly application of the biasing WFST and the off-line composition of the biasing model applied as a first-pass LM.

Table 2 shows the results for off-line composition of the models and using the composed model as the first-pass LM (i.e., the G of the CLG). As expected, there is a benefit to performing off-line composition of the biasing WFST, because some pruning (though not all) is performed prior to on-the-fly rescoring, relying solely on the first-pass model. On-the-fly composition loses 0.4% absolute of the total 1.8% WER reduction in the full two-pass biased recognizer.

Our presented methods can be applied to cases where this off-line CLG construction would be infeasible: contexts which change frequently (e.g., per utterance) or which are too detailed and numerous to permit separate full models for each. The two constraints which we are working with are time (required to perform composition before model can be used) and space (required to store and serve the potentially many large models).

3.4. Analysis

To get a sense of how biasing impacts specific recency-related examples, we selected utterances that have the following n-grams in the recognized transcripts: *'Charlie'* and *'winter storm J'* from dates January 17 and 27, 2015. These are dates where search queries for *'Charlie Hebdo'* and *'winter storm Juno'* peaked respectively. We re-recognized these utterances using on-the-fly rescoring with a recency biasing WFST. Table 3 captures some of the common misrecognitions we have without recency biasing. Overall, with the biasing WFST, we see a 10% increase in the number of utterances with the hypothesis *'Charlie Hebdo'*, 32% increase for *'je suis Charlie'* and 589% increase for *'winter storm Juno'*.

4. Related Work

N-gram model adaptation most commonly takes the form of model mixing or MAP adaptation (see [2] for a presentation of such methods for multinomial models in general), whereby

N-gram	Incorrect recognition examples
Charlie Hebdo	*Charlie Abdo*, *Charlie had to go* cover, French magazine *Charlie Eppes*, who is *Charlie Hipp do*
je suis Charlie	*Jay sweet Charlie*, *jesli Charlie*, translate *just sweet Charlie*
winter storm Juno	pictures of *winter storm Juneau*, power outages *winter storm Juneau* east coast, who named *winter storm Juneau*

Table 3: Incorrect recognition examples of n-grams without recency biasing. With recency biasing, the italicized n-grams are properly replaced with the n-gram on left.

in-domain and out-of-domain models are mixed based on parameters derived for each history that occurs in either model. In such approaches, to ensure proper normalization, the parameters driving the mixture are defined for each history.

A more complicated method, which combines differently for different n-grams with the same history, is the "fill-up" method [8]. In this approach, a discounted relative frequency is taken from the in-domain model if it has been observed in the domain; otherwise, if the n-gram has been observed in the out-of-domain model, the out-of-domain model score is used. If the n-gram has been observed in neither domain, the in-domain backoff probability is used. Everything is properly normalized.

Our approach is most similar to cache models [1], which are typically mixed with the baseline model with a fixed interpolation weight to preserve normalization. Several improvements to the original formulation, including a decay function [9] and generalization to higher order n-grams [10] have made this a potentially effective mode of personalization. Our approach contrasts with these methods by (1) deriving our salient n-grams from other sources besides personal language use; and (2) permitting more complex methods of combination with the baseline model, sometimes at the expense of normalization in the interests of run-time efficiency.

5. Conclusions

We have presented methods for collecting contextually salient n-grams, deriving target costs for those n-grams and efficiently combining on-the-fly during the first-pass of recognition. Results making use of *biasing WFSTs* derived from fast rising queries in search traffic demonstrate strong improvements in WER on utterances with high expectation of containing a fast rising query, while remaining neutral on general voice search traffic. We demonstrate the importance of biasing probabilities only in the case that the biasing results in a lower cost for the n-gram than the baseline probability, i.e., it is only promoting n-grams based on the sample, not inhibiting n-grams.

There remain open questions for making use of the myriad of potential contextual signals available at time of recognition, such as how to combine two independent biasing WFSTs, e.g., from fast rising queries and also from some kind of geo-location information. The importance of positive biasing in the current case gives us some indication of one promising approach to this: choose the maximum positive bias from among the set. N-gram selection and bias estimation also require further research.

6. Acknowledgements

Thanks to Petar Aleksic, Ciprian Chelba, Babak Damavandi, Mohamed Elfeky, Pedro Moreno, Johan Schalkwyk, Trevor Strohman, and Xuedong Zhang for extensive discussion.

7. References

[1] 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.

[2] M. Bacchiani, M. Riley, B. Roark, and R. Sproat, "MAP adaptation of stochastic grammars," *Computer speech & language*, vol. 20, no. 1, pp. 41–68, 2006.

[3] B. Roark, R. Sproat, C. Allauzen, M. Riley, J. Sorensen, and T. Tai, "The OpenGrm open-source finite-state grammar software libraries," in *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics, 2012, pp. 61–66.

[4] K. Wu, C. Allauzen, K. Hall, M. Riley, and B. Roark, "Encoding linear models as weighted finite-state transducers," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

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

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

[7] 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.

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

[9] 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 International Conference on*, vol. 2. IEEE, 1997, pp. 799–802.

[10] J. T. Goodman, "A bit of progress in language modeling," *Computer Speech & Language*, vol. 15, no. 4, pp. 403–434, 2001.

[11] P. Aleksic, M. Ghodsi, A. Michaely, C. Allauzen, K. Hall, B. Roark, D. Rybach, and P. Moreno, "Bringing contextual information to google speech recognition," in *in Proc. Interspeech*, 2015.