February 08, 2018

Contextual Speech Recognition Using Rescore-Aware Pruning

Ian Williams

Petar Aleksic

CONTEXTUAL SPEECH RECOGNITION USING RESCORE-AWARE PRUNING

ABSTRACT

A method for rescore aware pruning in speech recognition is disclosed. The method includes determining a set of possible words W_P that are reachable in a CLG state from an active hypothesis H_I. The CLG state may be mapped and dynamically assigned to CL state for determining W_P and then remapped to the CLG. Given this information, a backtrace state associated with H_i in the CLG is examined and the set of words W_R that will be rescored is determined. If W_P and W_R share an element, i.e., $W_P \cap W_R \neq \emptyset$, then H_I is retained and not pruned. Further, an additional beam may be used for increasing search space and include additional rescore-hypotheses.

Keywords: Speech decoding, speech recognition, context, lexicon and grammar (CLG), first-pass language model, rescore-aware pruning, additional beam method

BACKGROUND

In speech recognition, spoken words or utterances are decoded by a device to interpret content information. A major challenge in speech recognition is to accurately decode the speech and avoid misrecognition of words for improved user experience. Incorrectly decoded speech can be frustrating to users where no amount of biasing allows the decoder to recognize biased words. For example, in traffic-related search, biasing of nearby locations might fail and an establishment name may be misrecognized. Such an instance may be a mild annoyance. However, for must-recognize situations, such as when telling a smart speaker device to lower the volume or in a situation involving a Yes/No/Cancel dialog state, an error may leave behind a very bad experience for the user.

Rescoring is an approach to enable real-time accurate speech decoding. On-the-fly rescoring models provide an excellent mechanism to adapt language models for contextual signals that are unavailable ahead of time. However, the contextual information cannot be used in the first-pass search over the CLG finite state transducer (FST). This results in certain hypotheses being dropped due to their perceived unlikeliness, when they may be very likely for a given utterance and context. Such errors can be fixed when rescore-aware pruning is applied.

DESCRIPTION

Rescore-aware pruning refers to a technique whereby hypotheses in the decoder are spared from the normal pruning discipline if they lie along the path of a word that may eventually be rescored. The rescoring models may influence the pruning behavior of the first-pass search and allow certain hypotheses to be retained based on information unavailable to the first-pass language model (LM).

Given a particular transition into a state in a finite-state transducer based on context, lexicon and grammar (CLG) from an active hypothesis H_I, a set of possible words W_P are to be determined that are reachable from that state. Given this information, we examine the backtrace state associated with H_I and find the set of words W_R that will be rescored. If $W_P \cap W_R \neq \emptyset$, then H_I is retained, provided its score is within the rescore threshold S_R of the current best hypothesis' score, H_B. A detailed method for rescore-aware pruning in speech recognition is illustrated in FIG. 1 and described below.

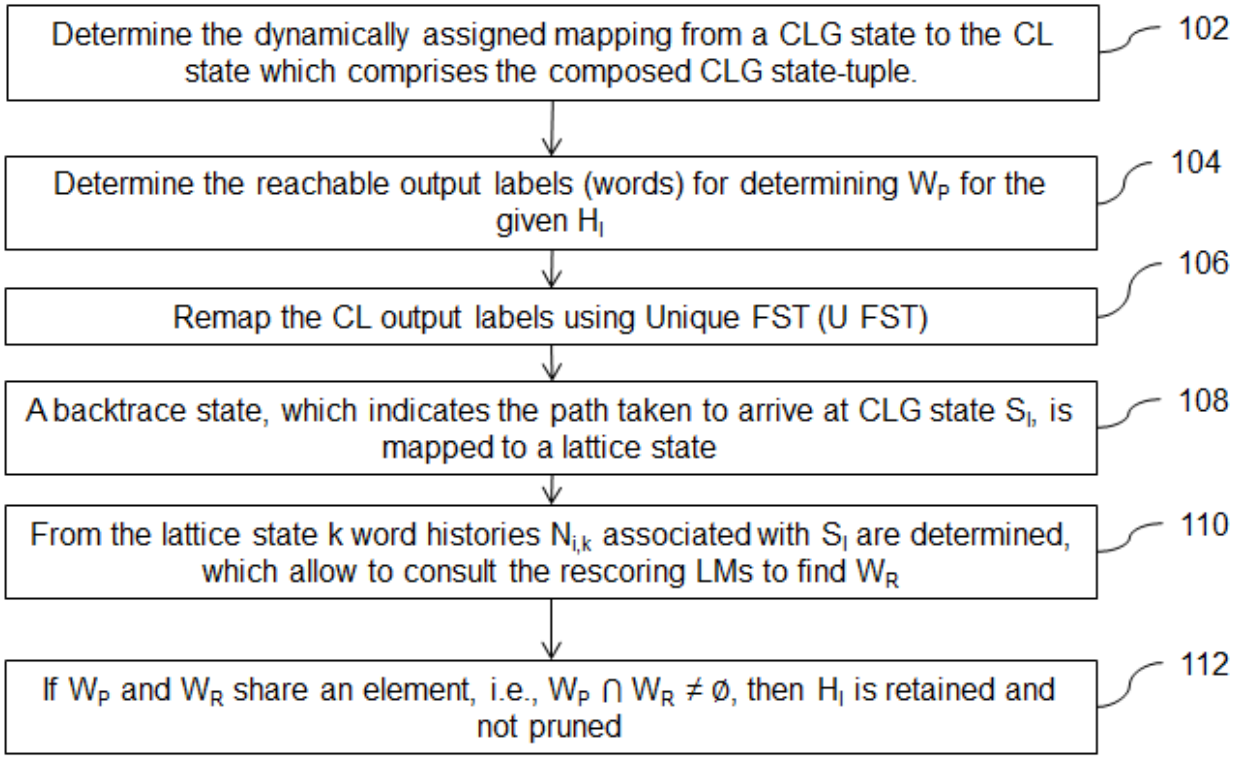

FIG. 1: Method for rescore-aware pruning for contextual speech recognition

The CLG is a gigantic data-structure and so is composed dynamically as needed. Hence, there is no fixed mapping from a CLG state to a collection of words reachable by that state. ComposeStateTable may be used to determine the dynamically assigned mapping from a CLG state to a context and lexicon (CL) state, which comprises the composed CLG state-tuple, at block 102. The context and lexicon (CL) FST is much smaller than the CLG and is realized with less computation. It is also an OLabelLookaheadFst, so the reachable output labels (words) may be determined, given any state in the CL FST. These features in the CL state enable determination of set of possible words W_P for the given hypothesis H_I, at block 104.

The permissiveness of the search for possible words with respect to keeping additional rescore-hypotheses is affected by three parameters. Since the CL is conceptually very much like a prefix tree, all words W are reachable at the start and fewer words become reachable as more

transitions are taken. This property means that little weight is likely to fall on the early transitions, when W_P is very large. Hence, it makes sense to consider keeping hypotheses only for CL states where W_P is below some threshold, which is the first parameter. The second parameter is a predetermined choice of rescore threshold, S_R, which is the maximum score difference allowed between a hypothesis that meets the rescore condition and the best hypothesis, H_B. If S_R is too low, then good hypotheses may not be retained and if it is too high then search resources are wasted on useless hypotheses. Third, a maximum number of rescore hypotheses may be set. If there are a huge number of hypotheses allowed by the rescoring condition, S_R must be reduced until it falls below this maximum limit, to prevent the search space from exploding.

The determined CL output labels do not directly correspond to the G output labels as there can be many pronunciations for the same word. The mapping for these many-to-one labels is given by the U FST ("unique"), which is used for re-mapping from the CL state to the CLG state, at block 106. For any H_I at CLG state S_I, there is an associated backtrace state that indicates the path taken to arrive at S_I. This backtrace state is mapped by some data structure to a lattice state (the details depend on the lattice implementation), at block 108. From the lattice state, the k word histories $N_{I,K}$ associated with S_I are determined. The word histories allow consultation of the rescoring LMs to find W_R, at block 110. If W_P and W_R do not share any element, i.e., $W_P \cap W_R \neq \emptyset$, then the H_I is retained and not pruned, at block 112.

The above described method may be implemented by examining, at the time that a hypothesis H_I is to be pruned, the CL look-ahead data structure and retrieving W_P. This data is stored in memory and may be used in-place throughout. The backtrace for H_I is consulted and the best cost word history $N_{I,0}$ is used to find a subset of W_R, [illegible]. Now, in order to check if $W_P \cap$

$\square_{\square}^{\square} \neq \emptyset$, it is determined if they share an element. This may be done by determining which sets are worth sorting at runtime in order to get the best performance in this shared-element check. One way to improve this process is to cache, per N, the set S_R of all states that have been found to be kept. This cuts out the computation of $W_P \cap \square_{\square}^{\square} \neq \emptyset$ for all but the first time, at the cost of increased memory use. This implementation is plug-and-play, so a rescoring LM does not need to do anything and its words will already be incorporated into a rescore-aware search. Further, there are no additional data structures needed. However, this implementation is slow and expensive. Additionally, despite caching and optimizations, this implementation does not work fast enough.

The present method may also be implemented by building ahead of time a mapping from output label (word) W_I to CL states C_I which lay in the path from the start state to the arc that outputs W_I. This data structure is loaded once into memory when the decoder is initialized, and is shared thereafter. This is a relatively large data structure, on the order of $|W|S_W$, where W is the set of all recognized words (~4 million), and S_W is the mean number of states leading to a word in the CL. S_W may be limited as needed and tuned as appropriate. When a rescoring LM is constructed at the beginning of an utterance decode, it creates its own mapping from word history N to all states lying along paths reaching W_R by consulting pre-built mapping. As mentioned in the earlier implementation, only the best cost word history $N_{I,0}$ may be used for our hypothesis H_I. The complexity here is not bad so long as the rescoring LM is reasonable in size. This must be done only once per decode. If there are multiple rescoring LMs acting during a decode, the number of potential word history permutations may be very large and sparsely visited by the rescoring logic. Therefore, the multiple rescoring LM will take the sets of states S_W for each word W that is rescored by one of the underlying rescorers and create a merged set.

The merged set is cached to avoid frequent re-creation. This implementation is very fast and involves little time overhead. It allows any rescoring LM to specify to the decoder which hypothesis to consider more thoroughly in a general fashion. However, it requires additional data structures to be built offline and uses additional memory.

Based on the present method, two-channel audio test sets were generated. On the latest audio history two-channel test sets, Table 1 and Table 2 represent the differences generated on the largest test sets. The test sets are flat to slightly positive, with some showing 0.1 WER reduction and up to 0.5% SACC increase.

Table 1: Audio history of noisy two-channel audio test sets

Gain	Ref	New_Hyp	Old_Hyp
IMPROVEMENT	Mute the microphone	mute the microphone	Newton microphone
IMPROVEMENT	Where can I find a…in San Francis	where can I find a…in	where can I find a…
IMPROVEMENT	Volume 2	volume two	phone to
REGRESSION	File feedback	file a feedback	file feedback
REGRESSION	Add volume	raise volume	Add volume
REGRESSION	Little quieter	quieter	Little quieter

Table 2: Audio history two-channel audio test sets

Gain	Ref	New_Hyp	Old_Hyp
IMPROVEMENT	Play sneezing panda on video website…	Play sneezing panda on…	play Stevie Wonder on…
IMPROVEMENT	Pause the TV	pause the TV	has the TV

IMPROVEMENT	Mute the microphone	mute the microphone	Newton microphone
IMPROVEMENT	Where can I find a…in San Francis	where can I find a … in	where can I find a …
IMPROVEMENT	Volume 2	volume two	phone to
REGRESSION	File feedback	file a feedback	file feedback
REGRESSION	Add volume	raise volume	Add volume
REGRESSION	Little quieter	quieter	Little quieter

From the above results, it may be observed that the method fixes many visible commands (pause, volume, etc), fixes a number of "Play sneezing panda videos ….". A small sample size shows 2^{-4} impact, which may be attributed to raters incorrectly categorizing a significant number of wins as audio 'not present' due to speaker being far from mic, or raters categorizing many differences as losses incorrectly because of ambiguity, or re-running with latest tuning parameters and instructions for raters to be aware that this is an audio beacon device and be aware of media commands and far away speakers.

In theory these same gains may be achieved, as shown in Table 1 and 2, by increasing the search space using the normal beam. A test set of 50 utterances from a smart speaker that are fixed (in-part or in whole) using additional beam was constructed by examining the above results and observations, as shown in Table 3. A sweep of beam size and max_arcs was performed over these utterances to match the performance of the additional beam method. This was done with both the rescored token set lattice and the newer word lattice, with their default operating points.

Table 3: Results for increased search space based on additional beam

	WER	SACC	Average States	Average Arcs
Baseline (no additional beam, default settings)	63.8	0	355k	1,255k
Baseline w/ additional beam (beam 18, rescore arcs 40)	21.6	66	357.5k	1199k
Baseline w/ additional beam (beam 22, rescore arcs 100)	12.15	80	367k	1180k
Best (no additional beam, beam 17, arcs 8000, token set lattice)	12.9	78	1066k	3,980k

Further, the present method may be augmented by using first-pass LM scores to adapt the pruning threshold. For instance, by looking up the first-pass LM score while creating the rescoring model and using that value as a dynamic rescore threshold, S_R could prevent wasting some computation on useless rescore paths. Additionally, the method could also consider a word history other than the best scoring one, which may cause a given hypothesis to be rescored.