A GENERAL ALGORITHM FOR WORD GRAPH MATRIX DECOMPOSITION

Dilek Hakkani-Tür and Giuseppe Riccardi[*]

AT&T Labs-Research,
180 Park Avenue, Florham Park, NJ, USA
{dtur,dsp3}@research.att.com

ABSTRACT

In automatic speech recognition, word graphs (lattices) are commonly used as an approximate representation of the complete word search space. Usually these word lattices are acyclic and have no a-priori structure. More recently a new class of *normalized* word lattices have been proposed. These word lattices (a.k.a. sausages) are very efficient (space) and they provide a normalization (*chunking*) of the lattice, by aligning words from all possible hypotheses. In this paper we propose a general framework for lattice chunking, the pivot algorithm. There are four important components of the pivot algorithm. First, the time information is not necessary but is beneficial for the overall performance. Second, the algorithm allows the definition of a predefined chunk structure of the final word lattice. Third, the algorithm operates on both weighted and unweighted lattices. Fourth, the labels on the graph are generic, and could be words as well as part of speech tags or parse tags. While the algorithm has applications to many tasks (e.g. parsing, named entity extraction) we present results on the performance of confidence scores for different large vocabulary speech recognition tasks. We compare the results of our algorithms against off-the-shelf methods and show significant improvements.

1. INTRODUCTION

In large vocabulary continuous speech recognition (LVCSR), the word search space, which is prohibitively large, is commonly approximated by word lattices. Usually these word lattices are acyclic and have no a-priori structures. Their transitions are weighted by acoustic and language model probabilities. More recently a new class of *normalized* word lattices have been proposed [1]. These word lattices (a.k.a. sausages) are more efficient than canonic word lattices and they provide an alignment for all the strings in the word lattices.

In this paper we propose a general framework for lattice chunking, the pivot algorithm. In terms of state transition matrix this corresponds to decomposing the lattice transition matrix into a block diagonal (chunk) matrix. Figure 1 shows the state transition matrices for topologically sorted traditional lattices and the new type of lattices we propose, the pivots. The elements $a_{i,j}$ are binary, 1 if there is at least one transition between states i and j, 0 otherwise. In the rest of the paper we will refer to $a_{i,j}$ as the equivalence class of state transitions from state i to j. The state transitions $a_{i,j}$ can be weighted or unweighted. In the weighted case, the cost associated to the transition from state i to state j, with label w_k is $c_{i,j}^{w_k}$.

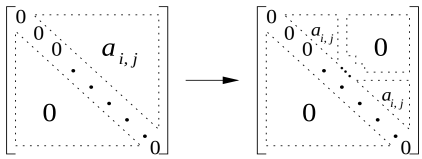

Fig. 1. The state transition matrices for topologically sorted traditional lattices and pivot alignments. $a_{i,j}$ is 1 if there is at least one transition between states i and j, 0 otherwise.

There are four important components of the pivot algorithm:

1. The time information computed using the frame numbers is not necessary but is beneficial for the overall performance.
2. The algorithm allows the definition of a predefined chunk structure for the final lattice.
3. The algorithm operates on both weighted and unweighted lattices.
4. The labels on the graph are generic and could be words as well as part of speech tags or parse tags.

We describe the algorithm here in the context of automatic speech recognition (ASR). Lattice chunking has the clear advantage of *normalizing* the search space of word hypotheses. The advantages of these normalized lattices are in terms of memory and computation:

- **Memory** The resulting structures (pivots) are much smaller in size (order of magnitudes), while preserving accuracy of the original search space.

[*] The authors are listed in alphabetical order.

- **Computation** Normalized matrices have the *compositional property*. Suppose that we want to compute the word string with lowest weight, C_{min} among all strings $W = w_1, \ldots, w_k, \ldots, w_N$ in the lattice:

$$C_{min} = \min_{W} \sum_{w_k \in W} c_{i,j}^{w_k} = \sum_{m_l} \min_{W_l} \sum_{v_k} c_{i,j}^{v_k} \quad (1)$$

where $W_l = v_1, \ldots v_k, \ldots v_M$ is the set of word strings recognized by the l_{th} lattice chunk, m_l.

There are many applications where these properties have been very useful. In the case of weighted lattices, the transition probabilities on the pivots can also be used as word confidence scores. The posterior probabilities on the most probable path of the resulting pivot alignment have been used as confidence scores for unsupervised learning of language models [2] and active learning for ASR [3]. The pivot structure of competing word hypotheses, as well as their confidence scores have been used for improving spoken language understanding [4], machine translation [5] and named entity extraction [6]. In [7] the compositional property has been extended to the case of weighted string costs. In this paper we present the application of the pivot algorithm to the computation of word confidence scores for all the strings in a word lattice. We present results on the performance of confidence scores for a large vocabulary continuous speech recognition task.

In the next section, we describe the algorithm. In the third section, we provide experimental results.

2. APPROACH

The sausage algorithm proposed in [1] is designed to reduce word error rate and is thus biased towards automatic speech recognition. The pivot algorithm is general and aims to *normalize* the topology of any input graph according to a canonic form. The parameters of the algorithm can be used to optimize a specific cost function (e.g., word error rate). The algorithm is summarized in Figure 2 and a brief description of the steps is given below:

1. If the lattice is weighted, we first compute the posterior probability of all transitions in the word graph, by doing a forward and a backward pass through the graph. At this point, the posterior probability of a transition could be used as a confidence score by itself, but some improvements are possible by taking into account the competing hypotheses in the same time slot. In the case of unweighted lattices, we skip this step.

2. We then sample a sequence of states that lie on a path,[1] in the lattice, to use as the baseline of the pivot alignment. This path can be the best path or the longest path of the lattice, as well as any random path. The selection of the path can be optimized towards a specific cost function (e.g., word error rate). In most of our experiments, we either used the best or the longest path. The states on the pivot alignment baseline are assumed to inherit their time information from the lattice. In our algorithm, the time information is not necessary, but beneficial for the overall performance. We define time slot $ts(T)$ of transition T as the speech interval between the starting and ending time frames of T.

3. In the lattice, each transition overlapping $ts(T)$ is a *competitor* of T, but competitors having the same word label as T are *allies* [8]. We sum the posterior probabilities of all the allies of transition T and we obtain what we call the posterior probability of word w. To compute the sum of the posterior probabilities of all transitions labeled with word w, that correspond to the same instance, we traverse the lattice in topological order, and insert all transitions into the pivot alignment baseline. When we find the most overlap-

[1] A path is a sequence of state transitions from the initial state to a final state.

1. Compute the posterior probabilities of all transitions T in the word graph.
2. Extract the pivot baseline path.
3. For all transitions T in the topologically ordered lattice, do:

1. Using $ts()$, find the most overlapping location on the pivot baseline (defined by a start state S_s and an end state S_e).
2. If there is no transition at that location that precedes T in the lattice,
   (a) If a transition with the same label already occurs at that location, add posterior probability of T to the posterior probability of that transition.
   (b) Otherwise, insert a new transition to that location with the label and posterior probability of T.
3. Otherwise,
   (a) Insert a new state S_n to the pivot alignment.
   (b) Assign that state a time information.
   (c) Change the destination state of all transitions originating from state S_s to S_n.
   (d) Insert a transition between states S_n and S_e, assign it the label and posterior of T.

Fig. 2. The pivot algorithm.

ping location on the baseline, defined by a source and a destination state, we check if there is already a transition at that location that precedes T on a path in the lattice. Insertion of T at that location would violate the transition ordering defined by the initial lattice. If there is no such transition, we check if another transition with the same label already occurs in between those two states. In the presence of such a transition, we increment its posterior probability by the posterior probability of the new transition. In the absence of a transition with the same label, we create a new transition from the source to destination state, with the label and the posterior probability of the currently traversed transition on the lattice. If the insertion of T violates the transition ordering of the lattice, we create a new location, by inserting a new state in between source and destination. We change the destination state of all the transitions from the source state and make them point to the newly inserted state. We insert the current transition from the lattice, in between the newly created state and the destination state. In the current implementation, we assign the newly inserted state, the mean of the times of source and destination states as state time.

When the time information is not available, we assign each state of the lattice its approximate location on the overall lattice. According to this, the initial state is assigned a location 0, the final states that do not have any outgoing transition are assigned a location 1. All the other states in between are assigned a real number in $(0, 1)$, obtained by dividing the average length of all paths up to that state by the average length of all paths that go through that state. These numbers can be computed by a forward and a backward pass through the lattice. We use these approximate state locations to obtain $ts(T)$. The pivot algorithm runs in $O(n \times k)$ time, where n is the number of state transitions in the lattice, and k is the number of chunks in the resulting structure plus the average fan-out of the pivot alignment states at the time of the insertion. k is usually much less than n. For example, if the best path is used as the pivot baseline, then k is the length of the best path plus the number of state insertions made and the average fan-out. The complexity of the pivot algorithm is better than the algorithm of Mangu *et. al.* which runs in $O(n^3)$ time.

3. EVALUATION

We performed a series of experiments to test the quality of the pivot alignments and the confidence scores on them. For these experiments, we used a test set of 2,174 utterances (31,018 words) from the database of the *How May I Help You?*$^{\mathrm{SM}}$ (*HMIHY*^{SM}) system for customer care [9]. The language models used in all our experiments are trigram models based on Variable Ngram Stochastic Automata [10]. The acoustic models are subword unit based, with triphone context modeling and variable number of gaussians (4-24). The word accuracy of our test set when recognized with these models is 66.2%, and the oracle accuracy of the output lattices is 85.7%. Oracle accuracy is the word accuracy of the path in a lattice, whose labels are closest to the reference sequence. It is an upper-bound on the word accuracy that can be obtained using these lattices. To assess the quality of confidence scores on pivot alignments, we plot false rejection versus false acceptance. False rejection (FR) is the percentage of words that are correctly recognized in the ASR output, but are rejected as their confidence score is below some threshold. False acceptance (FA) is the percentage of words that are misrecognized but are accepted as their confidence score is above that same threshold. In Figure 3, we have plotted the FR versus FA curves using different thresholds, for four different types of confidence scores: the posterior probabilities of the transitions on the best path of the lattice, the most likely path of the pivot alignments using approximate time information, consensus hypotheses of sausages, and the most likely path of the pivot alignments using time information. The curve that is closest to the origin is the best one, as it has the minimum error rate (false rejection and acceptance). Both pivot alignments and sausages result in better confidence scores than the naive approach of using the posterior probabilities on the best path of the lattice. Although the pivot alignments using time information were generated in much less time than sausages, their FR versus FA curve is almost overlapping with the one obtained using sausages [1]. When the time information is not available, the FR versus FA curve for the pivot alignments is only slightly worse than the one obtained using time.

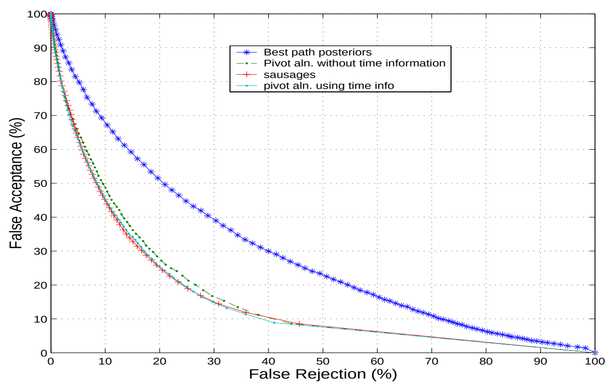

Fig. 3. FR versus FA curves for various confidence scores.

Another method for testing the quality of the confidence scores is checking the percentage of correctly recognized words for given confidence scores. One may expect k% of the words having the confidence score of $k/100$ to be correct. Figure 4 shows our results for confidence scores extracted from the best path of pivot alignments computed

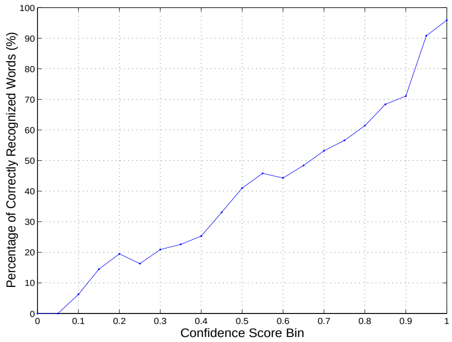

Fig. 4. Percentage of correctly recognized words in confidence score bins.

k	**Oracle Accuracy**
0.4	67.5%
0.2	70.8%
0.1	74.2%
0.05	77.0%
0.01	81.2%
0	86.7%

Table 1. Oracle accuracies when we pruned all transitions having a posterior probability less than k.

without using time information. As seen, the percentage of correctly recognized words in each confidence score bin increases almost linearly as the confidence score increases.

To assess the quality of the pivot alignments, we have computed oracle accuracies after pruning the pivot alignments with two different criteria. In Table 1, the oracle accuracies after pruning the pivot alignments by using a threshold for posterior probability are presented. Any arc which has a posterior probability less than k has been pruned from the pivot alignment, then the oracle accuracy has been computed on the pruned pivot alignment. In Table 2, the oracle accuracies after pruning the pivot alignments using the rank of the transitions are presented. In between all two states connected by a transition, only the top l transitions that have the highest posterior probability has been retained when computing the oracle accuracy. For example, if we use only the two transitions that have the highest posterior probabilities, we can achieve an oracle accuracy of 75.5%. These numbers indicate that, using the top candidates in the pivot alignments, instead of just the ASR 1-best hypothesis, it is possible to be more robust to ASR errors.

The sizes of the pivot alignments are much smaller than the corresponding lattices. In our tests, the size of the pivot alignments is 7% of the size of the lattices.

l	**Oracle Accuracy**
1	66.2%
2	75.5%
3	79.0%
4	80.9%
∞	86.7%

Table 2. Oracle accuracies when we took only the most probable l candidates in between all states.

4. CONCLUSIONS

We have proposed a general algorithm for lattice chunking. Our algorithm does not require any time information on the input lattice, and the labels of the lattice can be words as well as part of speech tags or parse tags. While the algorithm has applications to many tasks, such as parsing and named entity extraction, we described the algorithm in the context of ASR. We have presented the application of the algorithm to the computation of word confidence scores. We have compared the results of our algorithm against off-the-shelf methods and have shown significant improvements.

5. REFERENCES

[1] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, no. 4, pp. 373–400, 2000.

[2] R. Gretter and G. Riccardi, "On-line learning of language models with word error probability distributions," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, 2001, pp. 557–560.

[3] D. Hakkani-Tür, G. Riccardi, and A. Gorin, "Active learning for automatic speech recognition," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2002, pp. 3904–3907.

[4] G. Tur, J. Wright, A. Gorin, G. Riccardi, and D. Hakkani-Tür, "Improving spoken language understanding using word confusion networks," in *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, 2002, pp. 1137–1140.

[5] S. Bangalore, G. Bordel, and G. Riccardi, "Computing consensus translation from multiple machine translation systems," in *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2001.

[6] F. Bechet, J. Wright, A. Gorin, and D. Hakkani-Tür, "Named entity extraction from spontaneous speech in How May I Help You?^{SM}," in *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, 2002, pp. 597–600.

[7] S. Kumar and W. Byrne, "Risk based lattice cutting fir segmental minimum bayes-risk decoding," in *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, 2002, pp. 373–376.

[8] D. Falavigna, R. Gretter, and G. Riccardi, "Acoustic and word lattice based algorithms for confidence scores," in *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, 2002, pp. 1621–1624.

[9] A. Gorin, J.H. Wright, G. Riccardi, A. Abella, and T. Alonso, "Semantic information processing of spoken language," in *Proc. of ATR Workshop on Multi-Lingual Speech Communication*, 2000, pp. 13–16.

[10] G. Riccardi, R. Pieraccini, and E. Bocchieri, "Stochastic automata for language modeling," *Computer Speech and Language*, vol. 10, pp. 265–293, 1996.