# IMPROVED CONFUSION NETWORK ALGORITHM AND SHORTEST PATH SEARCH FROM WORD LATTICE

*Jian Xue and Yunxin Zhao*

Department of Computer Science
University of Missouri, Columbia, MO 65211 USA
jxwr7@mizzou.edu zhao@cs.missouri.edu

**ABSTRACT**

In this work, we propose a novel confusion network(CN) generation algorithm with linear time complexity O(T), which is capable of transforming a very large lattice into a confusion network with insignificant time. We further extend the confusion network concept to incorporate the case that a long word is split into short words. Finally, we develop an algorithm of shortest path search that finds a sentence hypothesis from a word lattice to directly minimize expected word error rate. The proposed algorithms are evaluated on the Switchboard task, where significant reduction of computation time was observed for the proposed confusion network algorithm as compared with a previously proposed confusion network algorithm, and improved word accuracy performance was observed for both the proposed CN algorithm and the shortest path algorithm as compared with one-best beam search decoding.

## 1. INTRODUCTION

The goal of standard maximum a posterior probability (MAP) speech decoder is to find the sentence hypothesis that maximizes the posterior probability P(W|A) of word sequence W given an acoustic observation A. Such an approach is known to minimize sentence error rate. However, commonly used performance metric in speech recognition is word error rate. There is therefore a mismatch between decoding criterion and performance measurement.

Many state-of-the-art speech recognition systems provide word lattice output in addition to the MAP-based sentence hypothesis W for each speech utterance. A word lattice is a Directed Acyclic Graph (DAG) which contains a large number of competing word hypotheses, denoted by corresponding links and their associated likelihood scores. The link scores are obtained through a combination of acoustic and language model probabilities. Mangu et al [1] proposed transforming word lattice into confusion network (CN), and utilized CN for producing sentence hypothesis that minimizes expected word error rate. The CN algorithm of [1], referred to as MBS-CN, successfully reduced word error rate. However, its time complexity is high. For a lattice with T links, the time complexity of MBS-CN is $O(T^3)$. The high complexity of MBS-CN was alleviated by aggressive link pruning in word lattice. Since in an online system, the overall processing time, including front-end processing, feature analysis, lattice generation, hypothesis alignment and best hypothesis extraction, needs to be less than 1.0×real time, it is desirable to investigate more efficient methods for confusion network generation, or expected word error rate minimization.

In the current work, we propose a novel confusion network (CN) generation algorithm with linear time complexity O(T). The proposed algorithm is capable of transforming a very large lattice into a confusion network with insignificant time. We further extend the confusion network concept to accommodate the case that a long word gets split into short words. Finally, we develop an algorithm of shortest path search that finds a sentence hypothesis directly from a word lattice to minimize expected word error rate. The proposed algorithms were evaluated on the Switchboard task, where significant reduction of computation time was observed for the proposed confusion network algorithm as compared with MSB-CN, and improved word accuracy performance was observed for both the proposed CN algorithm and the shortest path algorithm over conventional one-best beam search.

The rest of the paper is organized as the following. Section 2 introduces the basic concepts of confusion network. Section 3 presents the fast CN algorithm. In section 4 the shortest path search algorithm is derived. In section 5 experimental results of these methods based on the Switchboard 2001 HUB-5 Corpus are presented. We conclude our work in the final section 6.

## 2. BASICS OF CONFUSION NETWORK

The confusion network as proposed in [1] aligns links on a word lattice and transforms the lattice into a linear graph in which all paths pass through all nodes. An example of confusion network is shown in Fig. 1. In Fig. 1, each pair of adjacent nodes defines a position in the CN.

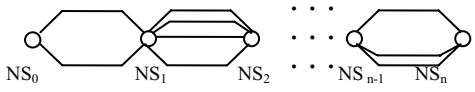



Fig. 1 Example of confusion network

The transformation is performed by a clustering procedure that groups time overlapped links into clusters based on their phonetic similarity and word probabilities while preserving the

This work is supported in part by National Institutes of Health under the grant NIH 1 R01 dc04340-01A2 and the National Science Foundation under the grant NSF EIA 9911095.

precedence order of the links encoded in the original lattice. The confusion network was utilized in [1] to generate word sequence hypothesis that minimizes expected word error rate rather than sentence error rate, where the word error criterion has a better match with the Levenshtein-distance based performance metric in speech recognition. Given the alignment and the link posterior probabilities of a confusion network, the word sequence hypothesis with the lowest expected word error is obtained by picking the word with the highest posterior probability at each position in the alignment. The main complexity of the confusion network approach lies in finding an optimal multiple string alignment of reference sentence hypotheses with a word sequence hypothesis, which has no known efficient solution. Here we propose a novel confusion network generation algorithm with the linear time complexity O(T).

## 3. THE PROPOSED FAST CN ALGORITHM

The nodes in a confusion network, e.g., Fig. 1, are in effect node sets, with each node including a set of nodes in the original lattice. It is desired to divide the nodes in the lattice into a finite number of sets $N_0$, $N_1$, . . . $N_n$, so that the start and end nodes of each link in the original lattice are put into two consecutive node sets. Strictly, not all DAGs can be transformed into confusion network as defined. In order to generate CN with O(T), we resort to the following heuristic approach.

**Assumptions:**
Let N={$n_0$, $n_1$, . . .} be the set of nodes and E={$e_0$, $e_1$, . . .} be the set of links in the original lattice, where every node $n_i \in N$ has a time mark $t(n_i)$. Let $e_{u->v}$ denote a link in the lattice with the start and end nodes u and v. Let *NS*={ $N_0$, $N_1$, . . .} be the set of node sets in the confusion network and let $E_{N_i->N_j}$ be the set of links with start and end node sets $N_i$ and $N_j$. The following properties are assumed for the confusion network:

a. $\forall\, n_i \in N_i$ and $n_j \in N_i$, $t(n_i)<t(n_j) \Rightarrow i \le j$

b. $\forall\, n_i \in N_i$ and $n_j \in N_i$, $t(n_i)=t(n_j) \Rightarrow i=j$

c. $\forall\, e_{u->v} \in E$, if for $u \in N_i$ and $v \in N_j$, $e_{u->v}$ corresponds to a link set $E_{N_m->N_n}$, then $i \le m < n \le j$ (here $n=m+1$, $N_i$ and $N_j$ are not consecutive, so we should align $e_{u->v}$ to two consecutive node sets $N_m$ and $N_n$.)

**Algorithm description:**
As in the MBS-CN algorithm, the fast CN algorithm requires calculation of the posterior probability for each link in the lattice. The link posterior probability P(*l*| A) is defined as the sum of the probabilities of all paths passing through the link *l* normalized by the sum of probabilities of all paths, which can be computed by the forword-backword algorithm[1]. The fast algorithm consists of the following steps.

Step-1 Compute link posterior probability for each link in the lattice.
Step-2 Sort all nodes in N in order of increasing $t(n_i)$.
Step-3 Assign $n_0$ to $N_0$.
Step-4 For each node $n_i$, in the order of i=1, 2, . . .,

i) Suppose $n_{i-1}$ belongs to $N_j$. If there is no link between any node in $N_j$ and $n_i$, assign $n_i$ to $N_j$, otherwise assign $n_i$ to $N_{j+1}$.

ii) For every link $e_{u->n_i} \in E$, suppose u belongs to $N_s$ and $n_i$ belongs to $N_t$. If t =s+1, then the link is directly assigned to $E_{N_s->N_t}$. Otherwise, the link is assigned to $E_{N_{n-1}->N_n}$ with $s+1 \le n \le t$, where the link word probability and degree of time overlap are considered in the link placement, i.e., $n = \arg\max_{s+1 \le k \le t} \{SIM(E_{N_{k-1}->N_k}, e)\}$

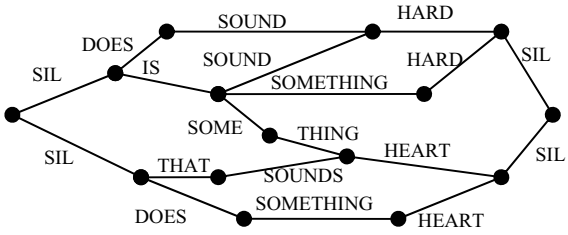
with

$$SIM(E_{N_{k-1}->N_k}, e) = \frac{1}{|E_{N_{k-1}->N_k}|} \times \sum_{l \in E_{N_{k-1}->N_k}} sim(w(l), w(e)) overlap(E_{N_{k-1}->N_k}, e)$$

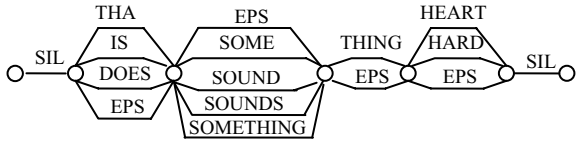
where w(*l*) and w(e) are words corresponding to *l* and e, sim(., .) is the phonetic similarity between two words, computed from the most likely phonetic base forms, overlap($E_{N_{k-1}->N_k}$, e) is defined as the time overlap between $E_{N_{k-1}->N_k}$ and e normalized by the sum of their lengths. The length of $E_{N_{k-1}->N_k}$ is $T_{max}(N_{k-1})$- $T_{min}(N_k)$, where $T_{max}(N_i) = \max\{t(n_j): n_j \in N_i\}$ and $T_{min}(N_i) = \min\{t(n_j): n_j \in N_i\}$

The time complexity of step 4 depends on the number of links in the set $E_{N_s->N_t}$. In general, even for a very large lattice, this number is less than 100. So the time complexity of step 4 is O(T). Furthermore, lattices as generated by various speech decoders, including HTK, have the nodes naturally sorted in order of increasing $t(n_i)$. Therefore, the time complexity of the proposed confusion network generation algorithm is O(T).

As a comparison in MBS-CN, the lattice to CN transformation is performed by a clustering procedure that groups time overlapping links into clusters while preserving the precedence order of the links, where $O(T^3)$ time is needed to initialize and preserve the order information.




(a) Initial lattice




(b) Confusion Network

Fig. 2. Transform a lattice into a Confusion Network

An example of transforming a word lattice into a confusion network by the proposed algorithm is shown in Fig. 2. In Fig. 2, EPS represents a NULL link, and its posterior probability equals 1 minus the sum of the posterior probabilities of the rest links at this position.

As is seen in Fig. 2(b), confusion network allows easy comparison of alternative word hypotheses for each position. The hypothesis with the lowest expected word error rate can be obtained by picking the word with the highest posterior probability at each position.

**Algorithm Improvement**

In a confusion network, each link sits between two consecutive node sets. If a link is very long, however, aligning the link to two consecutive node sets may not be reasonable. Fig. 3 illustrates such a case, where JULY is a much longer word than either the word DO or I.

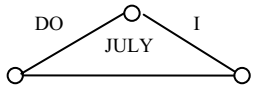



Fig. 3 Example of a long word broken into two short words

After forced alignment, one of the following two results will occur in the confusion network:


Fig. 4 Two possible alignments in confusion network for the case of Fig. 3

Based on the pronunciation and time information of these three words, we know that JULY should correspond to a concatenation of DO and I, that is, on one path the word output is JULY, and on another path the word output should be DO followed by I, which is a typical case of splitting a long word into two short words. To accommodate such a scenario and overcome the limitation in confusion network, we modify the algorithm such that when links are aligned, one link can end at two nonconsecutive node sets $N_i$ and $N_{i+2}$. The modified confusion network is illustrated in Fig. 5.

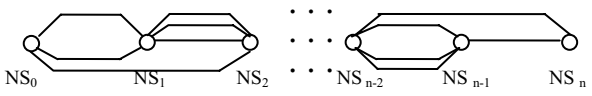



Fig. 5 Example of modified confusion network

If the case of Fig. 6 occurs for long links, the link with higher posterior probability is held fixed and the other one is then aligned between two consecutive node sets.

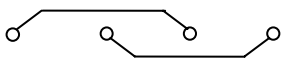


Fig. 6 The case prohibited in modified confusion network

In some cases, a long word may be split into three or more short words. For simplicity, the proposed algorithm only allows the case of one long word split to two short words.

The modification of the algorithm can be made in sub-step ii) of step 4. If t>s+1, the link e should be put in either $E_{N_{n-1}->N_n}$ or in $E_{N_{n-2}->N_n}$, based on link word probability and degree of time overlap. Computation of sim(., .) and overlap(., .) remain the same as before.

By applying the improved alignment algorithm, the lattice in Fig. 2(a) is transformed into the modified confusion network in Fig. 7.

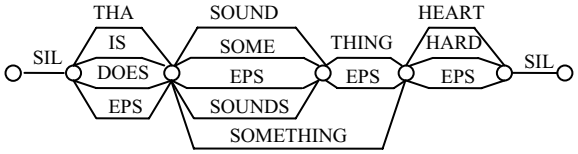



Fig. 7 Modified confusion network for the case of Fig. 2(a)

As pointed out in [1], confusion network has a variety of applications in speech recognition, in addition to minimization of expected word error rate. For example, the word-level posterior probabilities included in the confusion network can be conveniently used for confidence annotation of recognition output, i.e., estimating the probability of correctness of each word. For detailed discussion of other applications of CN, please refer [1].

## 4. SHORTEST PATH SEARCH

Minimization of expected word error rate can also be accomplished by a direct search on word lattice. To do so, we need to find a path with minimum word error rate, i.e., with minimum $E(\frac{1}{n} L(W,R))$, where n is the length of the reference string R. Given a word sequence $W = w_1 w_2 \ldots w_n$, L(W, R) is hard to compute because R is unknown. For simplicity, we assume W and R have the same number of words. We also make a strong assumption that $L(W, R) = \sum_{i=1\ldots n} L(w_i, r_i)$ [3], where $r_i$ is the ith word of R. Let $P(w_i \mid A)=P_i$. It is then easy to proof that

$$E(\frac{1}{n} L(W, R)) = 1-\frac{1}{n} \sum_{i=1\ldots n} P_i$$

Therefore, the path with minimum $1-\frac{1}{n} \sum_{i=1\ldots n} P_i$ needs to be determined. By defining the length of a path as the average length of its links and the length of a link as one minus the posterior probability of the corresponding word hypothesis, we can transform the decoding problem into finding a shortest path in the word lattice. Here the length of a path is the average length of its links, not the sum of the length of its links, as commonly defined in Graph Theory, so we cannot use any standard shortest-path graph algorithm.

**Algorithm description:**

Step-1. Use forward-backward algorithm to calculate link posterior probability for each link in the lattice;

Step-2. Topologically sort the nodes of the lattice as defined in section 3. For each node u, set $d_0(u)=0$, $d_i(u)=\infty$ and $\pi_i(u)= -1$. for i=1. . . m, where m equals the largest number of links of a path in the lattice;

Step-3. For each node u taken in topologically sorted order,

Do for each link $e_{v->u}$

Do for i=0 to m-1

If $\frac{d_i(v)*i+p(e_{v->u})}{i+1} > d_{i+1}(u)$

then set $d_{i+1}(u) = \frac{d_i(v)*i+p(e_{v->u})}{i+1}$

$\pi_{i+1}(u) = v$;

Step-4. Select n to maximize $d_n(u_e)$ for the end node $u_e$. Then n is the number of links of the shortest path.

Here a topological sort of a DAG G is a linear ordering of its nodes such that if G contains an edge $e_{u->v}$, then u appears before v in the ordering. As said above, nodes of word lattice are naturally sorted in order of increasing $t(n_i)$, so the sorting operation in step-2 is not needed, in general.

The node sequence $u_0=\pi_1(u_1), \ldots, u_{n-1}=\pi_n(u_n), u_n=u_e$ defines the best path that minimizing the expected word error rate $E(\frac{1}{n} L(W, R))$.

The time complexity of the shortest path search algorithm is also O(T), where T is the number of links in the lattice.

## 5. EXPERIMENTAL RESULTS

Experiments were conducted on the Switchboard 2001 HUB- 5 Corpus. HTK toolkit was employed to generate 310 word lattices for a test set of 310 sentences, by using triphone acoustic models and bigram language models. The acoustic models were provided by ISIP and language models by SRI.

Table 1. Comparison of execute time for word hypothesis generation with different lattice sizes.

| Method | Execution Time(× real time) Average number of links/lattice | | | | | |
|---|---|---|---|---|---|---|
| | 0-1k | 1-2k | 2-4k | 4-6k- | 6-8k | 8-10k |
| MBS-CN | <0.01 | 0.05 | 0.1 | 0.15 | 0.4 | 0.6 |
| Proposed-CN | <0.01 | 0.06 | 0.07 | 0.08 | 0.08 | 0.09 |
| Proposed shortest path | <0.01 | 0.02 | 0.02 | 0.03 | 0.03 | 0.03 |

A comparison of execution time of word hypothesis sequence generation among the proposed CN algorithm, the shortest path search algorithm and MBS-CN is shown in Table 1. The evaluation result of Table 1 was obtained by running the three algorithms on a 2GHz Pentium-4 processor and the results were averaged over the test set. Because MBS-CN was extremely slow, we pruned 95% of the links for MBS-CN, but pruned none for proposed CN and shortest path search. As such, the number of links for MBS-CN was actually 0.05×links/lattice for each case of links size in Table 1. From the table we observe that when the number of links per lattice was small, the three algorithms all run very fast. With the lattice growing larger, especially when number of links was larger than 6k, the MBS-CN algorithm required more than 0.4×real time. As a contrast, the proposed algorithms both required much less time, less than 0.1 real time.

Table 2. Comparison of decoding results of different methods.

| Method | Word Error Rate(%) |
|---|---|
| HTK's one-best | 47.57 |
| MBS-CN | 47.17 |
| Proposed-CN | 47.01 |
| Proposed shortest path | 47.30 |

Recognition word error rates were further compared across the three algorithms and the results are shown in Table 2. The results indicate that the expected word error rate minimization indeed yielded lower word error rate than sentence error minimization (the case of HTK's one-best), and the two confusion network methods were better than the shortest path method due to CNs' better utilization of competing path information in word lattice.

## 6. CONCLUSION

In this paper we proposed an improved confusion network algorithm that requires significantly less computation time and has a better capability of alignling long links as compared with the previously proposed MBS-CN algorithm. Wd also proposed a shortest path search algorithm that can produce word sequence hypothesis directly from word lattice with the property of minimizing expected word error rate. The effectiveness of the proposed techniques has been demonstrated on the Switchboard database.

## REFERENCE


[1] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minization and other application of confusion network," Computer Speech and Language, vol. 14 (4), pp. 373-400, 2000.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Introduction To Algorithms, The MIT Press, Cambrige, Massachusetts London, England

[3] Goel, V., Kumar, S., and Byrne, W., "Segmental minimum Bayes-risk decoding for automatic speech recognition," IEEE Trans. on Speech and Audio Processing, vol. 12 (3), pp. 234 -249, 2004.

[4] M. Weintraub, "LVCSR log-likelihood ratio rescoring for key word spotting," In proceedings of ICASSP, vol. 1, pp. 297-300, 1995