SYMBOL GRAPH BASED DISCRIMINATIVE TRAINING AND RESCORING FOR IMPROVED MATH SYMBOL RECOGNITION

Yu SHI and Frank K. SOONG

Microsoft Research Asia, Beijing, China
{yushi,frankkps}@microsoft.com

ABSTRACT

In the symbol recognition stage of online handwritten math expression recognition, the one-pass dynamic programming algorithm can produce high-quality symbol graphs in addition of the best recognized hypotheses [1]. In this paper, we exploit the rich hypotheses embedded in a symbol graph to discriminatively train the exponential weights of different model likelihoods and the insertion penalty. The training is investigated in two different criteria: Maximum Mutual Information (MMI) and Minimum Symbol Error (MSE). After discriminative training, trigram-based graph rescoring is performed in a post-processing stage. Experimental results finally show a 97% symbol accuracy on a test set of 2,574 written expressions with 43,300 symbols, a significant improvement of symbol accuracy obtained.

Index Terms— Handwritten math formula recognition, symbol recognition, symbol graph, discriminative training, graph rescoring

1. INTRODUCTION

In our previous work, a one-pass dynamic programming based symbol decoding and graph generation algorithm for online handwritten mathematical expression recognition was proposed [1]. It embeds segmentation into symbol identification to form a unified framework for symbol recognition. Besides the accurately recognized hypotheses, it can produce high-quality symbol graphs as well. During the decoding, there are six knowledge sources participating in the search. Similar to the language model scale factor and the insertion penalty adopted in speech recognition, exponential weights of different model likelihoods and the symbol insertion penalty should also be considered. In the previous system, the parameters are manually tuned. This process is very time and computation consuming. In this paper we would like to train them in an automatic way.

As having been proved in speech recognition, graph based discriminative training and post-processing are very useful to improve system performance. Along with the proposal of the framework for graph based discriminative training [2], it becomes more and more popular due to its efficiency and effectiveness. In this paper, we take full advantage of the symbol graph in improving the symbol recognition engine in an online handwritten math expression recognition system.

First, graph based discriminative training algorithm is proposed for the exponential weights of different model likelihoods and the insertion penalty, rather than for Hidden Markov Model (HMM) parameters in speech recognition. In comparison with the N-best list based discriminative training, graph based discriminative training is much more efficient due to the compact encoding method of alternative hypotheses. A framework for graph based discriminative training of HMM parameters could be found in Daniel Povey's thesis [2]. The Maximum Mutual Information (MMI) criterion showed its effectiveness in improving word error rates in Large Vocabulary Continuous Speech Recognition (LVCSR). Povey also presented a new discriminative training technique called Minimum Phone Error (MPE). This consistently gives better results than MMI, and appears to be a promising technique for discriminative training. In this paper, we focus on two training criteria: MMI and Minimum Symbol Error (MSE) whose concept is similar to MPE.

After the discriminative training of decoding parameters, symbol graph rescoring is applied in a post-processing stage. This provides an opportunity to further improve symbol accuracy by using more complex information that is difficult to be used in the one-pass decoding. In this paper, we rescore the symbol graph by introducing trigram syntax model probabilities.

2. SYMBOL DECODING ALGORITHM [1]

Under the assumptions that a user always writes a symbol without any insertion of irrelevant strokes before he finishes the symbol and each symbol can have at most of L strokes, a dynamic programming algorithm is feasible to be used in searching of an optimal symbol sequence within an affordable search space.

The goal of symbol decoding is to find out a symbol sequence $\hat{S}$ that maximize a posterior probability $P(S|O)$ given a sequence of input strokes $O = o_1 o_2 \cdots o_N$, over all possible symbol sequences $S = s_1 s_2 \cdots s_K$. Here K, which is unknown, is the number of symbols in a symbol sequence, and

s_k represents a symbol belonging to a limited symbol set Ω. As proposed in [1], two hidden variables are introduced into the search, which makes the Maximum A Posterior (MAP) objective function become

$$\hat{S} = \underset{B,S,R}{\arg\max}\, P(B,S,R|O) = \underset{B,S,R}{\arg\max}\, P(O,B,S,R) \qquad (1)$$

where $B = (b_0 = 0) < b_1 < b_2 < \cdots < (b_K = N)$ denotes a sequence of stroke indexes corresponding to symbol boundaries (the end stroke of a symbol), and $R = r_1 r_2 \cdots r_K$ represents a sequence of spatial relations between every two consecutive symbols. The second equal mark is satisfied because of the Bayes theorem.

By taking into account the knowledge sources used in [1], the MAP objective could be expressed as

$$\begin{aligned} P(O,B,S,R) &= P(O|B,S,R)P(B|S,R)P(S|R)P(R) \\ &= \prod_{k=1}^{K} \Big[P(o_i^{(k)}|s_k)P(o_g^{(k)}|s_k)P(o_r^{(k)}|r_k) \\ &\quad \times P(b_k - b_{k-1}|s_k)P(s_k|s_{k-1},r_k)P(r_k|r_{k-1}) \Big] \\ &= \prod_{k=1}^{K} \prod_{i=1}^{D} p_{k,i} \qquad (2) \end{aligned}$$

where $D = 6$ represents the number of knowledge sources in the search and the probabilities $p_{k,i}$ for i being 1 to 6 are defined as

$p_{k,1} = P(o_i^{(k)}|s_k)$	:	symbol likelihood
$p_{k,2} = P(o_g^{(k)}|s_k)$	:	grouping likelihood
$p_{k,3} = P(o_r^{(k)}|r_k)$	:	spatial relation likelihood
$p_{k,4} = P(b_k - b_{k-1}|s_k)$	:	duration probability
$p_{k,5} = P(s_k|s_{k-1}, r_k)$	:	syntax structure probability
$p_{k,6} = P(r_k|r_k - 1)$	:	spatial structure probability

A one-pass dynamic programming (DP) search of the optimal symbol sequence is then applied through the state space defined by the knowledge sources. Afterwards, single best results could be obtained. To generate symbol graph, we only need memorize all symbol sequence hypotheses recombined into each symbol hypothesis for each incoming stroke, rather than just the best surviving symbol sequence hypothesis in the single best method. For the search algorithm, a detailed description can be found in [1].

In speech recognition systems, there is an exponential weight between acoustic model likelihood and language model probability so as to make the system benefit from both on a certain scale (equalize and weight acoustic and language model likelihood contributions). In addition, an insertion penalty is always set to balance the insertion and deletion errors in the results. The traditional method to determine these parameters is to tune them on a development set to minimize the recognition error, since the parameters are few and manually tuning is easy. Soong *et al* proposed a Generalized Word Posterior Probability (GWPP) based method to search the optimal weights on the word verification error surface [3]. However, this is just feasible for low-dimensional search space.

In this paper, we assign different exponential weights to different model likelihoods and add an insertion penalty in symbol decoding to improve system performance. By doing this, the MAP objective in Equation (2) becomes

$$P_{\mathbf{w}}(O,B,S,R) = \prod_{k=1}^{K} \left(\prod_{i=1}^{D} p_{k,i}^{w_i} \times I \right) = \prod_{k=1}^{K} p_k \qquad (3)$$

where p_k is defined as a combined score of all knowledge sources and the insertion penalty for the k'th symbol in a symbol sequence

$$p_k = \prod_{i=1}^{D} p_{k,i}^{w_i} \times I \qquad (4)$$

w_i represents the exponential weights of the i'th model likelihood $p_{k,i}$ and I stands for the insertion penalty. The parameter vector needs to train is expressed as $\mathbf{w} = [w_1, w_2, \cdots, w_D, I]^T$.

In previous experiments in [1], we used a set of manually tuned parameters, while in this paper, we discriminatively train them on a training set.

3. DISCRIMINATIVE TRAINING

Discriminative training attempts to optimize the system performance by formulating an objective function that in some way penalizes parameter sets that are liable to confuse correct and incorrect answers. In this case, discriminative training requires a set of competing symbol sequences for one written expression. In order to speed up computation, the generic symbol sequences can be represented by only those that have a reasonably high probability. A set of possible symbol sequences could in principle be represented by an N-best list, that is, a list of the N most likely symbol sequences. A much more efficient way to represent them, however, is a symbol graph. This stores the alternative symbol sequences in the form of graph in which the arcs correspond to symbols and symbol sequences are encoded by the paths through the graph.

One advantage of using graphs is that the same graph can be used for each iteration of discriminative training. This separates the most time-consuming aspect of discriminative training, which is to find the most likely symbol sequences, and makes it only necessary to do once. This approach assumes that the initially generated graph covers all the symbol sequences that will have a high probability even given the parameters generated during later iterations of training. If this is not true, it will be helpful to regenerate graphs more than once during the training.

In this paper, we carry out discriminative training based on the symbol graphs generated through symbol decoding. There is no graph regeneration during the whole training procedure, that is, symbol graphs are used repeatedly.

3.1. Algorithm

Suppose that there are M training expressions. For training file $m, 1 \leq m \leq M$, let us denote the stroke sequence with O_m, the reference symbol sequence with S_m, and the reference symbol boundaries with B_m. No reference spatial relations are used here since we only care segmentation and symbol recognition quality. Hereafter, a symbol being correct means both its boundaries and symbol identity being correct, while a symbol sequence being correct indicates all symbol boundaries and identities in the sequence being correct. Assume S, B and R to be any possible symbol sequence, symbol boundary sequence and spatial relation sequence, respectively. Probability calculations in the training are carried out with probabilities scaled by a factor of κ. This is important if discriminative training is to lead to good test-set performance [4].

Two training criteria are adopted in this paper, they are Maximum Mutual Information (MMI) and Minimum Symbol Error (MSE). In objective optimization, the quasi-Newton method is used to find local optimal of the functions. Therefore, derivative of the objective with respect to each parameter must be produced. All these objectives and derivatives can be efficiently calculated via the Forward-Backward algorithm [5] based on symbol graph.

3.1.1. MMI criterion

The MMI training was proposed as a discriminative training criterion which would maximize the mutual information between the training symbol sequence and the observation sequence. Its objective function can be expressed as a difference of joint probabilities

$$\mathcal{F}_{\text{MMI}}(\mathbf{w}) = \sum_{m=1}^{M} \log \frac{\sum_R P_{\mathbf{w}}(O_m, B_m, S_m, R)^\kappa}{\sum_{B,S,R} P_{\mathbf{w}}(O_m, B, S, R)^\kappa} \tag{5}$$

Probability $P_{\mathbf{w}}(O, B, S, R)$ is defined as in (3). The MMI criterion equals the posterior probability of the correct symbol sequence, that is

$$\mathcal{F}_{\text{MMI}}(\mathbf{w}) = \sum_{m=1}^{M} \log P_{\mathbf{w}}(B_m, S_m | O_m)^\kappa$$

Substituting Equation (3) into (5), we have

$$\mathcal{F}_{\text{MMI}}(\mathbf{w}) = \sum_{m=1}^{M} \log \frac{\sum_R \prod_{k=1}^{K} p_{m,k}^\kappa}{\sum_{B,S,R} \prod_{k=1}^{K} p_k^\kappa} \tag{6}$$

where $p_{m,k}$ is the same with p_k except that the former corresponds to the reference symbol sequence of the m'th training data.

In the condition that all hypothesized symbol sequences are encoded by a symbol graph, the graph based MMI criterion can be formulated as

$$\mathcal{F}_{\text{MMI}}(\mathbf{w}) = \sum_{m=1}^{M} \log \frac{\sum_{U_m} \prod_{e \in U_m} p_e^\kappa}{\sum_U \prod_{e \in U} p_e^\kappa} \tag{7}$$

where U_m denotes a correct path in the graph for the m'th file, U represents any path in the graph, $e \in U$ stands for an edge belonging to path U, and p_e is the combined score with respect to edge e. By comparing equations (6) and (7), one can found that p_e and p_k are the same thing of different notations.

The denominator of Equation (7) is a sum of the path scores over all hypotheses. Given a symbol graph, it can be efficiently calculated by the Forward-Backward algorithm as $\alpha_0\beta_0$. While the nominator is a sum of the path scores over all correct symbol sequences. It can be calculated within the sub-graph G' constructed just by correct paths in the original graph G. Assume that the forward and backward probabilities for the sub-graph are α' and β', then the nominator can be calculated as $\alpha'_0\beta'_0$. Finally, the objective becomes

$$\mathcal{F}_{\text{MMI}}(\mathbf{w}) = \sum_{m=1}^{M} \log \frac{\alpha'_0\beta'_0}{\alpha_0\beta_0}$$

The derivatives of the MMI objective function with respect to the exponential weights and the insertion penalty can then be calculated as:

$$\begin{aligned}
\frac{\partial \mathcal{F}_{\text{MMI}}(\mathbf{w})}{\partial w_j} &= \sum_{m=1}^{M} \Big[\frac{\sum_{U_m} \prod_{e \in U_m} p_e^\kappa \sum_{e \in U_m} \log p_{e,j}^\kappa}{\sum_{U_m} \prod_{e \in U_m} p_e^\kappa} \\
&\qquad - \frac{\sum_U \prod_{e \in U} p_e^\kappa \sum_{e \in U} \log p_{e,j}^\kappa}{\sum_U \prod_{e \in U} p_e^\kappa} \Big] \\
&= \sum_{m=1}^{M} \Big(\frac{\sum_{e \in G'} \log p_{e,j}^\kappa \alpha'_e p_e^\kappa \beta'_e}{\alpha'_0\beta'_0} - \frac{\sum_{e \in G} \log p_{e,j}^\kappa \alpha_e p_e^\kappa \beta_e}{\alpha_0\beta_0} \Big) \\
\frac{\partial \mathcal{F}_{\text{MMI}}(\mathbf{w})}{\partial I} &= \sum_{m=1}^{M} \Big[\frac{\sum_{U_m} \prod_{e \in U_m} p_e^\kappa \sum_{e \in U_m} \kappa I^{-1}}{\sum_{U_m} \prod_{e \in U_m} p_e^\kappa} \\
&\qquad - \frac{\sum_U \prod_{e \in U} p_e^\kappa \sum_{e \in U} \kappa I^{-1}}{\sum_U \prod_{e \in U} p_e^\kappa} \Big] \\
&= \kappa I^{-1} \sum_{m=1}^{M} \Big(\frac{\sum_{e \in G'} \alpha'_e p_e^\kappa \beta'_e}{\alpha'_0\beta'_0} - \frac{\sum_{e \in G} \alpha_e p_e^\kappa \beta_e}{\alpha_0\beta_0} \Big)
\end{aligned}$$

In the derivatives, α_e and β_e indicate the forward and backward probabilities of edge e.

3.1.2. MSE criterion

The Minimum Symbol Error (MSE) criterion is directly related to Symbol Error Rate (SER) which is the scoring criterion generally used in symbol recognition. It is a smoothed approximation to the symbol accuracy measured on the output of the symbol recognition stage given the training data. The objective function in MSE, which is to be maximized, is:

$$\mathcal{F}_{\text{MSE}}(\mathbf{w}) = \sum_{m=1}^{M} \sum_{B,S} P_{\mathbf{w}}(B, S | O_m)^\kappa A(BS, B_m S_m) \tag{8}$$

where $P_{\mathbf{w}}(B, S | O_m)^\kappa$ is defined as the scaled posterior probability of a symbol sequence being the correct one given the weighting parameters. It can be expressed as

$$P_{\mathbf{w}}(B, S | O_m)^\kappa = \frac{\sum_R P_{\mathbf{w}}(O_m, B, S, R)^\kappa}{\sum_{B,S,R} P_{\mathbf{w}}(O_m, B, S, R)^\kappa} \tag{9}$$

$A(BS, B_m S_m)$ in Equation (8) represents the row accuracy of a symbol sequence given the reference for the m'th file, which equals the number of correct symbols

$$A(BS, B_mS_m) = \sum_{k=1}^{K} a_k, \quad a_k = \begin{cases} 1 & s_k, b_{k-1}, b_k \text{ are correct} \\ 0 & \text{otherwise} \end{cases}$$

The criterion is an average over all possible symbol sequences (weighted by their posterior probabilities) of the raw symbol accuracy for an expression. By expanding $P_{\mathbf{w}}(B, S|O_m)^\kappa$, Equation (8) can be expressed as:

$$\mathcal{F}_{\text{MSE}}(\mathbf{w}) = \sum_{m=1}^{M} \frac{\sum_{B,S,R} \prod_{k=1}^{K} p_k^\kappa A(BS, B_mS_m)}{\sum_{B,S,R} \prod_{k=1}^{K} p_k^\kappa}$$

Similar to the graph based MMI training, the graph based MSE criterion has the form

$$\mathcal{F}_{\text{MSE}}(\mathbf{w}) = \sum_{m=1}^{M} \frac{\sum_U \prod_{e\in U} p_e^\kappa \sum_{e\in U, e\in C} 1}{\sum_U \prod_{e\in U} p_e^\kappa} \tag{10}$$

where C denotes the set of correct edges. By changing the order of sums in the nominator, Equation (10) becomes

$$\mathcal{F}_{\text{MSE}}(\mathbf{w}) = \sum_{m=1}^{M} \frac{\sum_{e\in C} \sum_{U, e\in U} \prod_{e'\in U} p_{e'}^\kappa}{\sum_U \prod_{e\in U} p_e^\kappa} \tag{11}$$

The second sum in the nominator indicates the sum of the path scores over all hypotheses that pass e. It can be calculated from the Forward-Backward as $\alpha_e p_e^\kappa \beta_e$. The final MSE objective can then be formulated by the forward and backward probabilities as

$$\mathcal{F}_{\text{MSE}}(\mathbf{w}) = \sum_{m=1}^{M} \frac{\sum_{e\in C} \alpha_e p_e^\kappa \beta_e}{\alpha_0 \beta_0} \tag{12}$$

It equals the sum of posterior probabilities over all correct edges.

For the quasi-Newton optimization, the derivatives of the MSE objective function with respect to the exponential weights and the insertion penalty can be calculated as

$$\begin{aligned}
\frac{\partial \mathcal{F}_{\text{MSE}}(\mathbf{w})}{\partial w_j} &= \sum_{m=1}^{M} \Big[\frac{\sum_U \prod_{e\in U} p_e^\kappa \sum_{e\in U} \log p_{e,j}^\kappa \sum_{e\in U, e\in C} 1}{\sum_U \prod_{e\in U} p_e^\kappa} \\
&\quad - \frac{(\sum_U \prod_{e\in U} p_e^\kappa \sum_{e\in U, e\in C} 1)(\sum_U \prod_{e\in U} p_e^\kappa \sum_{e\in U} \log p_{e,j}^\kappa)}{(\sum_U \prod_{e\in U} p_e^\kappa)^2} \Big] \\
&= \sum_{m=1}^{M} \Big[\frac{\sum_{e\in C} \sum_{e'} \log p_{e',j}^\kappa \alpha_{e'}^{(e)} p_{e'}^\kappa \beta_{e'}^{(e)}}{\alpha_0 \beta_0} \\
&\quad - \frac{\sum_{e\in C} \alpha_e p_e^\kappa \beta_e}{\alpha_0 \beta_0} \frac{\sum_e \log p_{e,j}^\kappa \alpha_e p_e^\kappa \beta_e}{\alpha_0 \beta_0} \Big] \\
\frac{\partial \mathcal{F}_{\text{MSE}}(\mathbf{w})}{\partial I} &= \sum_{m=1}^{M} \Big[\frac{\sum_U \prod_{e\in U} p_e^\kappa \sum_{e\in U} \kappa I^{-1} \sum_{e\in U, e\in C} 1}{\sum_U \prod_{e\in U} p_e^\kappa} \\
&\quad - \frac{(\sum_U \prod_{e\in U} p_e^\kappa \sum_{e\in U, e\in C} 1)(\sum_U \prod_{e\in U} p_e^\kappa \sum_{e\in U} \kappa I^{-1})}{(\sum_U \prod_{e\in U} p_e^\kappa)^2} \Big] \\
&= \kappa I^{-1} \sum_{m=1}^{M} \Big[\frac{\sum_{e\in C} \sum_{e'} \alpha_{e'}^{(e)} p_{e'}^\kappa \beta_{e'}^{(e)}}{\alpha_0 \beta_0} \\
&\quad - \frac{\sum_{e\in C} \alpha_e p_e^\kappa \beta_e}{\alpha_0 \beta_0} \frac{\sum_e \alpha_e p_e^\kappa \beta_e}{\alpha_0 \beta_0} \Big]
\end{aligned}$$

Here $\alpha^{(e)}$ and $\beta^{(e)}$ indicate the forward and backward probabilities calculated within the sub-graph constructed by paths passing through edge e, while $\alpha_{e'}^{(e)}$ and $\beta_{e'}^{(e)}$ represents the particular probabilities of edge e'.

3.2. Experimental results

In this section, we experimentally investigate the discriminative training of exponential weights and insertion penalty. The database used in [1] is adopted here again. Symbol graphs are generated first by using the symbol decoding engine on the training data. Since MMI training must calculate the posterior probability of the correct paths, only those graphs with zero graph symbol error rate (GER) are randomly selected. The final data set for discriminative training has about 2,500 formulas, a comparable size with the test set. The graphs are then used for multiple iterations of MMI and MSE training. All parameters are initialized to 1 before the training.

3.2.1. Convergence

Fig. 1 shows the convergence of discriminative training with smoothing factor $1/\kappa = 0.3$. Both MMI and MSE objectives are monotonically increased during the process.

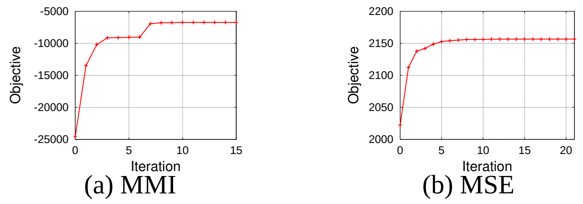

Fig. 1. Convergence property ($1/\kappa = 0.3$).

At each iteration of the training, the best path in the graph is investigated given the latest parameters. Both training and testing data are investigated. Fig. 2 shows the corresponding results with respect to symbol accuracy. In Fig. 2, (a) and (b) are obtained on training data, while (c) and (d) are obtained on testing data. We can observe that the improved performance can generalize to unseen data very well.

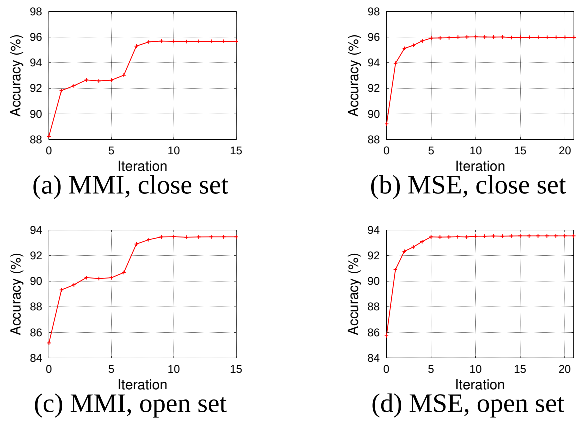

Fig. 2. Symbol accuracy of best path in graph ($1/\kappa = 0.3$).

system	accuracy	rel. impv.
two-step system	89.12%	-
bigram decoding (MMI)	93.33%	38.7%
bigram decoding (MSE)	93.41%	39.4%

Table 1. Symbol accuracy of bigram decoding.

3.2.2. Symbol accuracy

After discriminative training, we applied the obtained parameters in the symbol decoding engine to do a complete search. Table 1 shows the symbol accuracy and relative improvement obtained with different system configurations.

The first line in Table 1 illustrates the baseline results produced by the old system in which segmentation and symbol recognition are two separated steps. When comparing results of MMI and MSE training, we noticed that MSE training has achieved better performance than MMI training. This is consistent with Povey's conclusion in the MPE experiments in LVCSR. The reason is obvious. While the MMI criterion maximizes the posterior probability of the correct paths, the MSE criterion distinguishes all correct edges even in the incorrect paths. The MSE criterion has a closer relationship with the performance metric of symbol recognition, therefore, optimization of the MSE objective function will straight improve symbol accuracy.

4. GRAPH RESCORING

After discriminative training of the exponential weights and the insertion penalty, the system can be further improved by graph rescoring. In this paper, we introduce a trigram syntax model to the symbol graph so as to make the correct path more competitive. The trigram syntax model is formed by computing a probability for each symbol-relation pair given the preceding two symbol-relation pairs on a training set

$$P(s_k r_k | s_{k-2} r_{k-2}, s_{k-1} r_{k-1}) = \frac{c(s_{k-2} r_{k-2}, s_{k-1} r_{k-1}, s_k r_k)}{c(s_{k-2} r_{k-2}, s_{k-1} r_{k-1})}$$

where $c(s_{k-2}r_{k-2}, s_{k-1}r_{k-1}, s_k r_k)$ represents the number of times that triple $(s_{k-2}r_{k-2}, s_{k-1}r_{k-1}, s_k r_k)$ occurs in the training data and $c(s_{k-2}r_{k-2}, s_{k-1}r_{k-1})$ is the number of times that $(s_{k-2}r_{k-2}, s_{k-1}r_{k-1})$ is found in the training data. For triples that do not appear in the training data, smoothing techniques can be used to approximate the probability.

4.1. Graph expansion

From the definition of the trigram syntax model, it is required to distinguish both the last and second last predecessors for a given symbol-relation pair. Since the symbol-level recombination in the bigram decoding distinguishes partial symbol sequence hypotheses $s_1^k r_1^k$ only by their final symbol-relation pair $s_k r_k$, the symbol graph constructed in this way would have ambiguities of the second left context for each arc. Therefore, the original symbol graph must be transformed to a proper format before rescoring. Fig. 3 shows an example of the transformation. In comparison with the original graph, the transformed graph duplicated the central node so as to distinguish different paths recombined into the nodes at the right side.

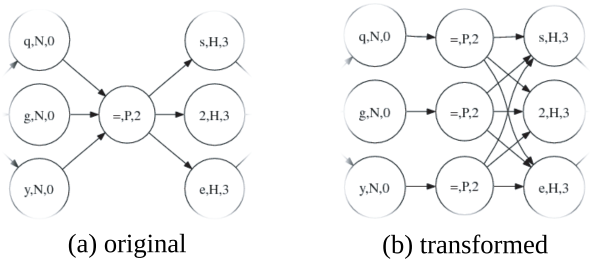

Fig. 3. Graph expansion.

4.2. Rescoring

After graph expansion, the trigram probability could be used to recalculate the score for each arc as follows

$$p_k = \prod_{i=1}^{D} p_{k,i}^{w_i} \times I \tag{13}$$

Here $D = 7$ rather than 6 in bigram decoding (Equation (4)), and $p_{k,7} = P(s_k r_k | s_{k-2} r_{k-2}, s_{k-1} r_{k-1})$ indicates the trigram probability. The exponential weight of the trigram probability still can be discriminatively trained together with the other weights and the insertion penalty based on the transformed symbol graph, in the same way as described in Section 3. Hence there are two sets of parameters in the system, one is of 6 dimensions and for bigram decoding and the other one is of 7 dimensions and for trigram rescoring.

4.3. Experimental results

In this section, we investigate the recognition performance achieved by graph rescoring. The new set of exponential weights and insertion penalty was also trained by both MMI and MSE criteria. After graph rescoring, the path with the highest score was extracted and compared with the reference to calculate the symbol accuracy. Table 2 shows the average symbol accuracy. Compared to the one-pass bigram decoding, the trigram rescoring got further significant improvement on symbol accuracy. The best result even exceeded 97%.

5. SUMMARY

The paper presented the use of discriminative criteria for training exponential weights and insertion penalty used in symbol decoding for handwritten math formula recognition. This

criterion	system	accuracy	rel. impv.
MMI	2-g decoding	93.33%	-
	3-g rescoring	96.94%	54.1%
MSE	2-g decoding	93.41%	-
	3-g rescoring	97.02%	54.8%

Table 2. Symbol accuracy of trigram rescoring.

includes the Maximum Mutual Information (MMI) and the Minimum Symbol Error (MSE) criteria. Both implementations of MMI and MSE training are carried out based on symbol graphs to represent alternative hypotheses of the training data. The quasi-Newton method was used for the optimization of the objective functions. Due to the Forward-Backward algorithm, the objectives and their derivatives were efficiently calculated through graph. Experiments showed that both criteria worked very well on unseen test data and produced significant improvement on symbol accuracy. Moreover, MSE reliably gave better results than MMI.

After discriminative training, graph rescoring was then performed by using a trigram syntax model. The graph was first modified by expanding nodes so that there will be no ambiguous path for trigram probability computation. Then arc scores are recomputed with the new probability. To do this, a new set of exponential weights and insertion penalty was trained based on the expanded graph. Experimental results showed dramatic improvement of symbol recognition through trigram rescoring.

In summary, via both graph based discriminative training and rescoring, the symbol recognition engine achieved a high performance of 97% in symbol accuracy.

6. REFERENCES

[1] Y. Shi, H. Y. Li, and F. K. Soong, “A unified framework for symbol segmentation and recognition of handwritten mathematical expressions,” in *ICDAR2007*, 2007, vol. II, pp. 854–858.

[2] D. Povey, “Discriminative training for large vocabulary speech recognition,” *Ph.D thesis*, July 2004.

[3] Soong F. K., Lo W.-K., and Nakamura S., “Optimal acoustic and langage model weights for minimizing word verification errors,” in *INTERSPEECH2004-ICSLP*, 2004, vol. TuB2003p, pp. 17–20.

[4] R. Schluter and W. Macherey, “Comparison of discriminative training criteria,” in *ICASSP1998*, 1998, pp. 493–496.

[5] S. Young *et al*, “The htk book (for htk version 3.3),” 2005.