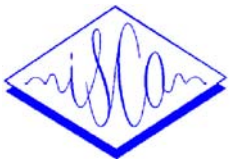

INTERSPEECH 2004 - ICSLP
8th International Conference on Spoken Language Processing
ICC Jeju, Jeju Island, Korea
October 4-8, 2004

DISCRIMINATIVE TRAINING OF COMPOUND-WORD BASED MULTINOMIAL CLASSIFIERS FOR SPEECH ROUTING

Xiang Li [1] *and Juan M. Huerta* [2]

[1] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213
[2] IBM T.J. Watson Research Center, Yorktown Heights, NY, 10598
xiangl@cs.cmu.edu, huerta@us.ibm.com

Abstract

We describe a method for utterance classification for speech routing based on a discriminative training multinomial topic classifier. We propose the utilization of the n-norm *a posteriori* topic probability of the speech hypothesis as the objective function of a discriminative training step, and explore various ways to compute and maximize this function with respect to model parameters. To avoid obtaining negative probability estimates, we propose an alternative representation of the model parameterization. We utilize our approach in combination with a simple non-discriminative word detection algorithm based on Mutual Information and with a technique to identify the most salient phrases of the domain in order to alleviate the feature sparsity problem. We also explore the post-processing of classification results to further improve the classification performance via a decision tree model and a neural network. Overall, our discriminative trained multinomial system reduces the classification error rate up to 45% in an NLU task of financial transactions comparative to our baseline non-discriminative multinomial system.

1. Introduction

Speech routing consists of selecting a topic from a set of topic candidates given a sentence or utterance. While the most obvious application of such classifiers is in the context of call routing in call center automation systems, other applications of this technique include support to topic-specific semantic parsing in NLU conversational systems [1]. Several techniques have been proposed for text routing (e.g., [2, 3]). As it has been discussed, a Maximum Likelihood approach to model estimation for call classification results in error rates which are higher than discriminative approaches. In this paper we propose a discriminative training method for a multinomial-based text classifier. Our technique maximizes an objective function based on the n-norm *a posteriori* class probability. We also discuss the issues of utilizing phrases or n-grams as features, identifying junk words (i.e., non-discriminative words) and post-processing topic classification output through Multi-layer perceptron neural network (MLP) and Decision Tree models. This paper is organized as follows: Section 2 describes the multinomial classification approach for utterance routing, Section 3 describes our proposed discriminative training method for the multinomial classifier, Sections 4 and 5 describe issues of n-gram feature selection and explicit feature pruning. In section 6 we present our experimental results. Post-processing of topic classification results through MLP and Decision Trees is discussed in section 7. Our paper concludes with the discussion in section 8.

2. Multinomial classification for call routing

In a multinomial-distribution (MN) text classifier each word W_i of a sentence $S = W_1, W_2, ..., W_N$ is treated as an independent observation of a multinomial distribution process, with distribution dependent on the topic T_k associated with the sentence S as follows:

$$P(S \mid T_k) = \frac{(\sum_i |W_i|)!}{|W_1|! \cdot |W_2|! \cdots |W_N|!} P_{W_1|T_k}^{|W_1|} \cdot P_{W_2|T_k}^{|W_2|} \cdots P_{W_N|T_k}^{|W_N|} \quad (1)$$

Where $|W_i|$ is the number of occurrences of word W_i, and $P_{W_i|T_k}$ is the probability of observing word W_i given topic T_k, which can be estimated via Maximum Likelihood (ML) criterion as in Equation 2:

$$[P_{W_i|T_k}] = Arg\max \prod_{S \in T_k} P(S \mid T_k) \qquad \{\sum_i P_{W_i|T_k} = 1\} \quad (2)$$

It turns out that the ML estimation of the term $P_{W_i|T_k}$ is:

$$\hat{P}_{W_i|T_k} = \frac{\sum_{T_k} |W_i|}{\sum_j \sum_{T_k} |W_j|} \quad (3)$$

where $\sum_{T_k} |W_j|$ is the number of occurrences of word W_j among all the sentences with topic T_k. For those words with zero occurrence in topic T_k, a probability floor can be applied.

3. Discriminative training of a multinomial classifier

One drawback of the Maximum Likelihood based Multinomial system (ML-MN) is that the model parameters are estimated only through the data of the true class itself, while the actual classification process requires the competition among different classes. In Discriminative Training (DT), model parameters are trained through the data not only from the true class, but also from all other competing classes. By incorporating competing information into the training process, a Discriminative Training Multinomial system (DT-MN) should perform better than the ML-MN baseline system.

3.1. Objective function P_s for sentence S

The first thing we need to do in DT-MN is to define an objective

function for individual training sentence $S = W_1, W_2, ..., W_N$. We thus define the objective function as the *n-norm a posterior* probability, as shown in equation (4) below

$$P_S = \frac{\prod_i P_{W_i|T_k} \cdot P_{T_k}}{[\sum_j (\prod_i P_{W_i|T_j} \cdot P_{T_j})^n]^{1/n}} \quad (4)$$

, where P_{T_j} is the prior probability of topic T_j, T_k is the true topic of sentence *S*, and *n* is the *n-norm* parameter. When *n* equals 1, our objective function is the exact *a posterior* probability of true topic T_k; when *n* is infinity, the term in the denominator is the topic with the highest joint probability.

3.2. Accumulation of the objective function across a training corpus

Once we define the objective functions for an individual training sentence, we need to accumulate the values of those objective functions across the training corpus. We propose three ways of accumulating objective functions: summation, multiplication (or log-summation) and Minimum Classification Error (MCE [3]) which employs a sigmoid function as in the following equations:

Summation: $P = \sum_S P_S$ (6)

Multiplication/log-summation: $LogP = \sum_S LogP_S$ (7)

MCE: $P = \sum_S L_S = \sum_S \frac{1}{1 + e^{-rP_S + \theta}}$ (8)

Where r and θ in equation (8) are the parameters that control the shape of sigmoid function curve.

3.3. Maximization of the objective function

We can use the gradient approach to optimize our objective functions; for this, we need the derivatives of those objective functions with respect to the model parameters. Depending on the way we accumulate the values of objective functions across training data, we obtain:

In Summation: $\nabla P = \nabla \sum_S P_S = \sum_S P_S \nabla LogP_S$ (9)

In Log-Summation: $\nabla LogP = \sum_S \nabla LogP_S$ (10)

In MCE: $\nabla P = \sum_S \frac{\partial L_S}{\partial P_S} \nabla P_S = \sum_S r L_S (1 - L_S) \cdot P_S \cdot \nabla LogP_S$ (11)

As can be seen from above equations, no matter how we accumulate the objective functions across training data, we need to compute the term $\nabla LogP_S$, which is the gradient of log objective function with respect to model parameters for sentence *S*. The closed-form expression for it turns out to be:

$$\nabla_{P_{W_i|T_k}} LogP_S = \frac{1}{P_{W_i|T_k}} [1 - \frac{(\prod_l P_{W_i|T_k} \cdot P_{T_k})^n}{\sum_m (\prod_l P_{W_i|T_m} \cdot P_{T_m})^n}] \quad (12)$$

for probability of word W_i given the true topic T_k, and

$$\nabla_{P_{W_i|T_j}} LogP_S = -\frac{(\prod_l P_{W_i|T_j} \cdot P_{T_j})^n}{\sum_m (\prod_l P_{W_i|T_m} \cdot P_{T_m})^n} \cdot \frac{1}{P_{W_i|T_j}} \quad (13)$$

for probabilities of word W_i given other competing topic T_j, where *n* is the *n-norm* parameter as in Equation (4).

Based on these solutions, we then iteratively update our model parameter $P_{W_i|T_j}$ using Equation (14) below:

$$P_{W_i|T_j}{}^t = P_{W_i|T_j}{}^{t-1} + \varepsilon \cdot \nabla P_{W_i|T_j}{}^{t-1} \quad (14)$$

One important thing to note is that since the parameters we are updating are actually probabilities, they should only be positive. In other words, the optimization process should be a constrained optimization, which can become quite elaborated. Although we can use simplifications of this constrained optimization (*e.g.* flooring the updated probabilities in each iteration), their performance is typically not satisfactory.

Alternatively, we can re-write our model parameter $P_{W_i|T_j}$ in terms of $C_{W_i|T_j}{}^{\alpha_{W_i|T_j}}$, and update the parameter $\alpha_{W_i|T_j}$ instead of $P_{W_i|T_j}$. Since $\alpha_{W_i|T_j}$ is the term in exponent, the updated model parameter $P_{W_i|T_j}$ will always be positive no matter what the value of $\alpha_{W_i|T_j}$ is. Another interesting point is that when $\alpha_{W_i|T_j}$ becomes 0 for all topics, the word W_i has effectively been *pruned out* as its presence in the sentence does not contribute (or affects) in any way the class posterior probability for any topic. Because of this reason, we call this method *soft-prune*, and the method that update the probabilities directly (equations 12-14) *linear-updating*. We can make the log base term $C_{W_i|T_j}$ constant across all words and topics.

It turns out that the closed-form expressions of the derivatives of log objective function with respect to the exponent $\alpha_{W_i|T_j}$ are:

$$\nabla_{\alpha_{W_i|T_k}} LogP_S = LogC_{W_i|T_k} [1 - \frac{(\prod_l C_{W_i|T_k}{}^{\alpha_{W_i|T_k}} \cdot P_{T_k})^n}{\sum_m (\prod_l C_{W_i|T_m}{}^{\alpha_{W_i|T_m}} \cdot P_{T_m})^n}] \quad (15)$$

for the exponents of the true topic T_k, and

$$\nabla_{\alpha_{W_i|T_j}} LogP_S = -LogC_{W_i|T_j} \cdot \frac{(\prod_l C_{W_i|T_j}{}^{\alpha_{W_i|T_j}} \cdot P_{T_j})^n}{\sum_m (\prod_l C_{W_i|T_m}{}^{\alpha_{W_i|T_m}} \cdot P_{T_m})^n} \quad (16)$$

for the exponents of the competing topic T_j. Similarly, we also use equation (14) in updating our exponents, with the term $P_{W_i|T_j}$ replaced by $\alpha_{W_i|T_j}$. Due to lack of space we omit the resulting expressions.

4. N-gram feature selection

Our ML-MN and DT-MN classifiers can be easily extended so they can include n-grams as features. However, the estimation of the model parameters can become imprecise if the feature space is sparse. Additionally, increasing the order of the features can substantially slow training and classification. In order to maintain a relatively small number of features while also considering groups of multiple words we propose the utilization of an n-gram feature selection technique.

In our n-gram feature selection process, we first compute a *stickiness* measure [4] $M(W,T)$ between topic T and each n-gram entry *W* as in equation (17):

$$M(W,T) = \sqrt{P(W|T) \cdot P(T|W)} = \frac{P(W,T)}{\sqrt{P(W) \cdot P(T)}} \quad (17)$$

then we sum the stickiness measures across all the topics as in equation (18)

$$M(W) = \sum_T M(W,T) \quad (18)$$

As features, we keep the top entries of the overall n-gram set (or more precisely, the set of n-grams of *up-to-n*th order) with largest sum of stickiness values to be our feature list. Alternatively, we

can pick the top-L entries of each topic and choose the union of those entries as the n-gram list.

When using the n-gram list, we decompose the likelihood of sentence $S = W_1, W_2, ..., W_N$ given topic T_k as in equation (19):

$$P(W_1, W_2, ..., W_N \mid T_k) = P(W_N \mid W_1, ..., W_{N-1}, T_k) \cdot P(W_{N-1} \mid W_1, ..., W_{N-2}, T_k) ... P(W_1 \mid T_k) \quad (19)$$

For each term $P(W_i \mid W_1, W_2, ..., W_{i-1}, T_k)$, we keep on backing off until we find the available entry in our n-gram list.

Estimating the n-gram term $P(W_i \mid W_1, W_2, ..., W_{i-1}, T_k)$ is the same as in the single word (uni-gram) case. We can either use ML estimation, with the constraints that the summation of the probabilities all word given the same left context and topic should be 1. We can also use DT to estimate n-gram probability.

5. Explicit word pruning

Not all the words that appear in the training corpus are useful, some are noise (*e.g.* "wow", "um"), or don't provide too much help in discriminating topics (*e.g.* "you", "I", "the"). In addition, our DT algorithms may get over-trained if there are too many *free* words in the vocabulary compared with the amount of training data. Because of these reasons, it would generally be beneficial if we can prune some words out of the vocabulary.

To achieve this, we could train the model first and then prune those noisy words using the cross-validation set. Instead, we propose a method to predict which words should be pruned before models are built; this type of algorithm is especially useful in the situation when there is no cross-validation set.

The first method we proposed is to use Mutual Information (MI) to decide which word should be pruned out. We first compute the MI between word W_i and topic T_k:

$$I(W_i, T_k) = \sum_{W_i, T_k} P(W_i, T_k) \cdot Log \frac{P(W_i, T_k)}{P(W_i) \cdot P(T_k)} \quad (22)$$

then accumulate the MI values across all topics for the same word as in equation (23):

$$I(W_i) = \sum_{T_k} I(W_i, T_k) \quad (23)$$

and finally delete those top N words with the least accumulated cross topic MI value in the vocabulary.

Another way of generating a junk-word list is based on the entropy measure of posterior probability of topics given a word as in Equation (24):

$$E_{W_i} = \sum_{T_k} P(T_k \mid W_i) \cdot Log \frac{1}{P(T_k \mid W_i)} \quad (24)$$

This method will prune those top N words with the maximal entropy value, which are words that provide minimum discriminative ability among topics.

6. Experimental results

We performed experiments using 3 datasets associated with a Natural Language based financial retirement planning telephony application. Those 3 sets differ from each other in terms of vocabulary, number of topics, and size. Table 1 shows the perplexity, vocabulary size, corpus size and topics per set.

	Perplexity	**Vocab. Size**	**Total/Unique training sents.**	**Number of Topics**
Set 1	8.12	1725	26362/12900	32
Set 2	8.12	1593	26362/12900	32
Set 3	4.37	1454	38084/9724	29

Table1: Characters of experiment databases.

First we evaluated the performance of our ML-MN and the DT-MN systems. We also tested the classification performance of a Decision Tree based Semantic Parser [1] in set 1 as comparison. Figure 1 illustrates the overall performance of various methods in terms of topic classification error rate, and figure 2 specifically lists the detailed results of various DT-MN configurations on Set1: columns labeled Parser and MN correspond the systems of decision tree based parser and ML-MN, all other items are as follows: the first letter specifies how the objective function has been accumulated (P: *multiplication*, S: *summation*, C: *MCE*), the second letter describes the way the derivative has been computed (L:*linear updating*, S: *soft-prune*); the last number tells the value of n in the *n-norm a posterior* probability. As we can see, the best DT configuration, corresponding to PS1, is 4.53% as reflected in both Figures 1 and 2 (vs. 8.18% for ML-MN).

Interestingly, among those misclassified sentences from our DT-MN system, on average 70% of them assign the true topic with the second or third highest probability. We carried out an n-best oracle experiment on test set 2 in which we treated a sentence as classified correctly as long as the true topic was among the top-3 highest probability topics. In that situation, the classification error rate went down to 5.7% from the previous 19.7% achieved by the DT-MN system. This suggests that we might get some additional performance gain by post-process the classification results, which will be discussed in the Section 6.

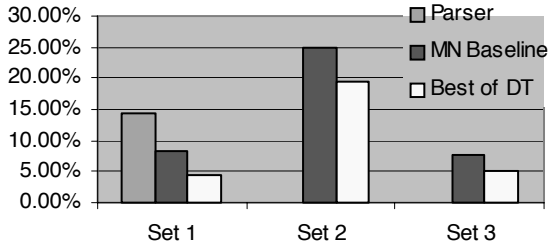

Figure1: Overall performance on different dataset. "Best of DT" corresponds to the best performance of DT-MN among its different configurations shown in Figure 2.

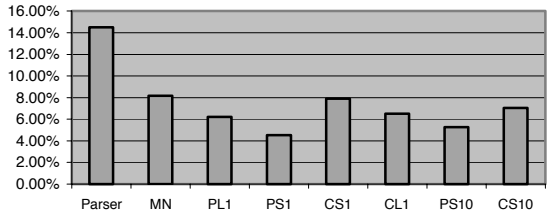

Figure2: Detailed DT-MN results for Set 1 using different configurations

We also tested the effects of using a unigram vs. a 4-gram feature set in set 3. The n-gram list is generated as the union of words with maximum *stickiness*. The experimental results are listed in Table 2.

Table 2 shows that the classification error rate on the training set is lower than the error rate on the testing set, which contradicts intuition. We believe this is due to the nature of the Set 3 dataset, which contains a lot of frequently appearing utterances in the training corpus. Misclassifications can be amplified due to this property. Another observation is that although the baseline performance of n-grams in the testing corpus is much better than

	Testing Set Error rates	Training Set	
		Error rates	Objective function log value
ML-MN (Baseline)	8.57%	12.8%	-16581
DT-MN (80 Iter.)	6.30%	9.69%	-8752
DT-MN (200 Iter.)	6.03%	9.35%	-7737
4-gram ML-MN	6.66%	8.32%	-7014
4-gram DT-MN	6.39%	7.80%	-4481

Table 2: Performance of n-gram model (n=4) and MN system (uni-gram only) on the Set 3 corpus.

the uni-gram the ML-MN baseline system, the N-gram DT-MN system is worse than the unigram DT-MN system. On the other hand, the error rate in the training set monotonically decreases as the value of objective function keeps up increasing (a similar trend is observed in [2]). We think that over-fitting may occur in the training set for the n-gram model, given the fact that there are many more free parameters in the n-gram model than in the uni-gram MN system.

Finally, we tested the effect of feature set pruning in Set 2 based on the Mutual Information criterion, which has better experimental performance than the entropy of the posterior probability criterion. In our experiments we only use the uni-gram configuration and we tested the performance of both the ML-MN baseline system as well as the DT-MN system. For the ML-MN system, the performance decreases monotonically as the number of words being pruned out increases. For the DT-MN system (illustrated in figure 3) we found some interesting results: if we look at the right most point, which corresponds to the situation where 1500 words have been pruned, its performance is still comparable with the case where no word has been pruned. We believe this tells us two things: 1st, there maybe a large redundancy of information provided by the words in the vocabulary; 2nd, DT is not only powerful, but also robust, especially when the number of free parameters it can utilize is limited.

7. Classification post-processing

As we mentioned in the previous section, the true topic in each sentence appears highly ranked in the n-best hypothesis list. Post-processing of the DT-MN system output may help us correct some of the error in those misclassified sentences.

We tried two post-processing approaches: the first was to train some Multi-Layer Perceptron (MLP) networks to perform an additional classification pass by using the probabilities of DT-MN system as input features and then combining those MLP decisions with the DT-MN decisions through majority voting; while the second one was to use a Decision Tree Model (DTM), which was trained from the ranked topic outputs in terms of the probabilities of the DT-MN system together with the probability of each topic normalized over the maximal probability.

Our post-processing experiments performed on Set 2 gave us some interesting results. In general, we achieved better classification accuracy using DTM, while not observing any substantial improvement from MLP post-processing. With DTM, we reduced the classification error rate in Set 2 from 19.8% to 18.7%. Using the MLP method, the topic classification error rate for the MLP itself was 20.1%, when combined with DT-MN via majority voting, the error rate was still 19.8%.

Despite the fact that majority voting of MLP and DT-MN didn't provide any substantial benefit, there was still some complementation in the performance of MLP and DT-MN output, which seems to suggest the potential advantage of

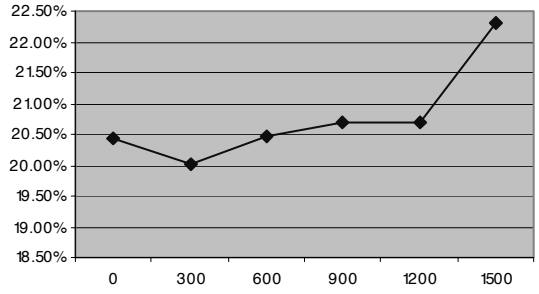

Figure3: The relation between the topic classification error rate of the DT-MN system and number of words been pruned in the Set 2.

hypothesis-combination. As reflected by those off-diagonal components in the confusion matrix of MLP and DT-MN in Table 3, in about 7% of the whole testing set, MLP and DT-MN disagree with each other, while one of them is correct. If we can develop a more judicious combination system to explore those off-diagonal components in the confusion matrix, we should get better combination performance.

Percentage (%)	DT-MN- correct	DT-MN Wrong
MLP Correct	76.4%	3.26%
MLP Wrong	3.82%	16.52%

Table 3: Confusion matrix of MLP and DT-MN system on the Set 3 corpus.

8. Conclusions

Discriminative training consistently reduced the error rate of our MN baseline system. It almost halved of original baseline error rate in one of our test sets. In addition, it's also robust, especially when the number of free parameters is limited.

N-gram and pruning are also very important issues that affect the topic classification performance, using them appropriately will improve the classification performance. Over-fitting is also a very important issues need to be considered accordingly.

Finally, post-processing is necessary and beneficial, by exploiting the error patterns and applying the corresponding error correction mechanism (*e.g.*, Decision Tree models or combination of various classification systems), the classification error rate may be reduced dramatically. Our experiment using Decision Tree models for post-processing partially supports this point.

9. Acknowledgements

We thank David Lubensky, Hong-Kwang Jeff Kuo and Cheng Wu for all the fruitful discussions related to this topic.

10. References

[1] Huerta et al., "Topic Specific Parser Design in an Air Travel NLU Application", *Eurospeech 2003*

[2] Chelba et al., "Discriminative Training of n-gram Xlassifiers for Speech and Text Routing", *Eurospeech 2003*

[3] Kuo et al., "Discriminative Training of Natural Language Call Routers" *IEEE Trans on Speech and audio processing, vol. 11, 2004*

[4] Saon et al., "Data Driven Approach to Designing Compound Words for Continuous Speech Recognition", *IEEE Trans on Speech and audio processing, vol. 9, 2001.*