SIMPLIFYING DESIGN SPECIFICATION FOR AUTOMATIC TRAINING OF ROBUST NATURAL LANGUAGE CALL ROUTER

Hong-Kwang Jeff Kuo and Chin-Hui Lee

Bell Labs, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974-0636, U.S.A.
email: {kuo,chl}@research.bell-labs.com

In this paper, we study techniques that allow us to relax some constraints imposed by expert knowledge in task specifications of natural language call router design. We intend to fully automate training of the routing matrix while still maintaining the same level of performance (over 90% accuracy) as that in an optimized system. Two specific issues are investigated: (1) reducing matrix size by removing word pairs and triplets in key term definition while using only single word terms; and (2) increasing matrix size by removing the need for defining stop words and performing stop word filtering. Since simplification of design often implies a degradation of performance, discriminative training of routing matrix parameters becomes an essential procedure. We show in our experiments that the performance degradation caused by relaxing design constraints can be compensated entirely by minimum error classification (MCE) training even with the above two simplifications. We believe the procedure is applicable to algorithms addressing a broad range of speech understanding, topic identification, and information retrieval problems.

1. INTRODUCTION

Touch-tone menus for routing callers are cumbersome to use when there are many destinations because the callers have to listen to all the choices and what they want to do may not even be obviously related to the given choices. Call routing based on spoken utterances have been proposed as an alternative that is natural and easier to use. However, callers may not know the name of the department they need (e.g. "home equity loan department") and only know what they want to do (e.g. "good morning um i wanna speak with someone about um a second mortgage loan do you all do that"). A previous study [1] has shown that callers prefer to specify the activity they need to accomplish rather than the name of the department, by a factor of more than three to one.

In natural language call routing, callers are routed to desired departments based on natural spoken responses to an open-ended "How may I direct your call?" prompt. [4, 1] In designing a voice response system to handle these calls, it is not sufficient to include in the speech recognizer just the names of the departments in the vocabulary or to use a finite state grammar, because what the callers may say cannot be fully anticipated. Instead, requests from real callers have to be collected for training the system. Data-driven techniques are therefore essential in the design of such systems.

In previous work [1], a vector-based information retrieval technique was introduced for performing call routing. A routing matrix was trained based on statistics of occurrence of words and word sequences in a training corpus after morphological and stop-word filtering. New user requests were represented as feature vectors and were routed based on the cosine similarity score with the model destination vectors encoded in the routing matrix.

In this paper we propose several techniques to simplify the training of the routing matrix in order to eliminate human expert knowledge in training the router for new domains. We simplify the features in order to reduce the number of parameters in the model and compensate for the resulting loss in performance by discriminative training [5]. In particular, we show that we are able to get performance similar to the original optimized system of over 90% accuracy. Discriminative training reduces the error rate by about 40%. Discriminative training increased the robustness of the classifier such that with 10% rejection, there was a relative error rate reduction of about 40%. The proposed formulation is applicable to algorithms addressing a broad range of speech understanding, topic identification, and information retrieval problems.

In the following sections, details of the proposed techniques and results of experiments will be presented.

2. VECTOR-BASED NATURAL LANGUAGE CALL ROUTING

In vector-based natural language call routing, call routing is treated as an instance of document routing, where a collection of labeled documents is used for training and the task is to judge the relevance of a set of test documents. Each destination in the call center is treated as a collection of documents (transcriptions of calls routed to that destination), and a new caller request is evaluated in terms of relevance to each destination. [6, 1, 2, 3]

The training process involves constructing a routing matrix R. Each document (customer utterances within a caller session) is first passed through morphological processing where the root forms of words are extracted. A list of ignore words are eliminated and a list of stop words are replaced with place holders. Then n-grams are extracted, specifically unigrams, bigrams and trigrams. Only unigrams that occur at least twice and bigrams and trigrams that occur at least three times in the corpus are included. This leads to a list of m terms (features).

The $m \times n$ term-document matrix is then constructed. The rows represent the m terms and the columns the n destinations. The routing matrix R is the transpose of the term-document matrix, where r_{vw} is the frequency with which term w occurs in calls to destination v. Each term is weighted according to term frequency inverse document frequency (TFIDF) and are also normalized to unit length.

New user requests are represented as feature vectors and are routed based on the cosine similarity score with the n model destination vectors $\vec{r}_i$ in the routing matrix R. Let $\vec{x}$ be the m-dimensional observation vector representing the weighted terms which have been extracted from the user's utterance. One possi-

ble routing decision is to route to the destination with the highest cosine similarity score:

$$\text{destination } \hat{j} = \arg\max_j \cos\phi_j = \arg\max_j \frac{\vec{r_j} \cdot \vec{x}}{\|\vec{r_j}\|\|\vec{x}\|}. \quad (1)$$

A classification error occurs when the score of the correct class is less than the maximum score. Notice that according to the way the routing matrix is constructed, there is no guarantee that the classification error rate will be minimized. The routing matrix can be improved by adjusting the models to achieve a minimum (at least locally, and in the probabilistic sense) of classification error rate, which is the subject of the next section.

3. MINIMUM CLASSIFICATION ERROR CRITERION

We had recently proposed using discriminative training to optimize the minimum classification error criterion for natural language call routing [5]. The same framework is used in this paper and is sketched out below.

The $n \times m$ elements of the routing matrix are regarded as the classifier parameters to be adjusted to achieve minimum classification error by improving the separation of the correct class from competing classes. The dot product of normalized query and destination vectors is used as the discriminant function. In the training algorithm, the model destination vectors are normalized after each adjustment step in order to maintain the equivalence between the measures of dot product and cosine score used in computing the classification error rate. Intuitively, this algorithm looks at each training example and adjusts the model parameters of the correct and competing classes in order to improve the scores of the correct class relative to the other classes.

Specifically, let $\vec{x}$ be the observation vector and $\vec{r}_j$ be the model document vector for destination j. We define the *discriminant function* for class j and observation vector $\vec{x}$ to be the dot product of the model vector and the observation vector:

$$g_j(\vec{x}, R) = \vec{r}_j \cdot \vec{x} = \sum_i r_{ji} x_i. \quad (2)$$

Note that this function is identical to the cosine score if the two vectors have been normalized to unit length.

Given that the correct target destination for $\vec{x}$ is k, we define the *misclassification function* as

$$d_k(\vec{x}, R) = -g_k(\vec{x}, R) + G_k(\vec{x}, R), \quad (3)$$

where

$$G_k(\vec{x}, R) = \left[\frac{1}{K-1} \sum_{j \neq k, 1 \leq j \leq K} g_j(\vec{x}, R)^\eta \right]^{\frac{1}{\eta}} \quad (4)$$

is the *anti-discriminant function* of the input $\vec{x}$ in class k and $K-1$ is the number of competing classes. Note that in the limit as the positive parameter $\eta \to \infty$, the anti-discriminant function is dominated by the biggest competing discriminant function: $G_k(\vec{x}, R) \to \max_{j \neq k} g_j(\vec{x}, R)$. Notice also that $d_k(\vec{x}, R) > 0$ implies misclassification, i.e. the discriminant function for the correct class is less than the anti-discriminant function.

A smooth differentiable 0-1 function such as the sigmoid function is chosen to be the *class loss function*:

$$l_k(\vec{x}, R) = l(d_k) = \frac{1}{1 + \exp^{-\gamma d_k + \theta}}, \quad (5)$$

where γ and θ are constants which control the slope and the shift of the sigmoid function, respectively.

The parameter set R is adjusted iteratively according to

$$R_{t+1} = R_t + \delta R_t. \quad (6)$$

where R_t is the parameter set at the t-th iteration. The correction term δR_t is solved using the training sample $\vec{x}_t$ given for that iteration, whose true destination is k:

$$\delta R_t = -\epsilon_t \nabla l_k(\vec{x}_t, R_t), \quad (7)$$

where ϵ_t is the learning step size.

Once this essential framework for discriminative training has been laid out, what is left is to work out the algebra for this particular problem. Let r_{vw} be elements of the routing matrix R. Then at iteration step t,

$$\nabla l_k(\vec{x}_t, R_t) = \frac{\partial l_k(\vec{x}_t, R_t)}{\partial R_t} = \frac{\partial l_k}{\partial d_k} \frac{\partial d_k(\vec{x}_t, R_t)}{\partial r_{vw}}. \quad (8)$$

Note that for the l_k we have chosen,

$$\frac{\partial l_k}{\partial d_k} = \gamma l_k(d_k)(1 - l_k(d_k)). \quad (9)$$

From equations 2, 3, and 4, the following can be shown:

$$\frac{\partial d_k(\vec{x}_t, R_t)}{\partial r_{vw}} = \begin{cases} -x_w & \text{if } v = k \\ \frac{x_w G_k(\vec{x}_t, R)(\vec{r}_v \cdot \vec{x}_t)^{\eta-1}}{\sum_{j \neq k} (\vec{r}_j \cdot \vec{x}_t)^\eta} & \text{if } v \neq k \end{cases} \quad (10)$$

Therefore, given the observation vector $\vec{x}_t$ at each iteration, each element of the routing matrix is adjusted according to:

$$r_{vw}(t+1) = \begin{cases} r_{vw}(t) + \epsilon_t \frac{\partial l_k}{\partial d_k} x_w & \text{if } v = k \\ r_{vw}(t) - \frac{\epsilon_t \frac{\partial l_k}{\partial d_k} x_w G_k(\vec{x}_t, R)(\vec{r}_v \cdot \vec{x}_t)^{\eta-1}}{\sum_{j \neq k} (\vec{r}_j \cdot \vec{x}_t)^\eta} & \text{if } v \neq k \end{cases} \quad (11)$$

Equation 11 shows that the model vector for the correct class is adjusted differently from those of the competing classes; notice in particular the difference in sign of the adjustment. Intuitively, the score of the correct class is improved relative to the scores of the competitors by the incremental adjustments. The adjustment to the wth component of each model vector is proportional to the learning step size ϵ_t, the size of the wth component in the observation vector $\vec{x}_t$, and the slope of the sigmoid function $\frac{\partial l_k}{\partial d_k}$. This slope is zero for very large or small values of d_k and positive in a certain region: the decision boundary which depends on θ and γ. Only the training data whose d_k values fall within the decision boundary will affect the model parameters significantly.

After each adjustment step, the affected models $\vec{r}_i$ are normalized to unit length in order that the discriminant function be identical to the cosine similarity score used in classification. The training vectors are normalized once before the discriminative training.

4. EXPERIMENTAL SETUP

Experiments were performed using the training and test sets collected for the USAA call routing task as reported in [1], consisting of a total of about 4000 calls. The same set of training data was used both to construct the initial routing matrix and for performing discriminative training. Each training vector is composed of the information provided by all the customer utterances within each

call session, including disambiguating follow-up utterances. Each call session has been manually routed to a destination, representing the ground truth of the correct class. There are a total of $n = 23$ destinations and $m = 756$ terms in the baseline system. In the discriminative training, multiple passes are made through the entire training set. Within each pass, the order in which each training vector is processed is randomized. For testing, only the 307 unambiguous initial utterances were used from the unseen test set. The baseline system as reported in [1] has a classification rate of 92.2% or error rate of 7.82% for this same set of 307 test utterances.

5. PARAMETER SELECTION

We see from the above equations that a number of parameters for GPD training have to be chosen. η controls the relative importance among the competitors – a larger value emphasizes the strongest competitors only. γ and θ controls the decision boundary through modifying the shape and location of the sigmoid function. ϵ_t controls the step size of the gradient descent. It is reduced gradually in order to achieve stable convergence; specifically, the step size is chosen to be a function like $1/t$, but chosen so that it changes only once every 25 passes. Note that K-1 is the total number of competitors to the correct class. In practice, the discriminative training can be focused on just the top M competitors instead of all $K-1$ classes. Another parameter is the number of passes through the training set which can be expressed as a stopping criterion, for example, when the change in the empirical loss function is less than a certain threshold.

A different set of paramters was used in this paper than the ones previously reported [5]. In the following results, we chose the following parameter values: $\eta = 20$, $\gamma = 32.0$, $\theta = 0.0$, $\epsilon_t = 3 \times 10^{-4}$ (initial), and $M = 6$. We found this setting improves the baseline performance previously reported in [5].

6. RESULTS

We first begin by presenting the baseline results of the natural language call router using 756 features, consisting of 62 trigrams, 274 bigrams, 420 unigrams. As shown in Table 1, the classification error rate was 7.8% and with discriminative training could be reduced to 5.5%, representing a relative improvement of 29%. With recognized strings from the speech recognizer (which has a word error rate of about 30%), the improvement is less, only about 12%.

	baseline	after DT	%change
human transcription	7.82%	5.54%	29%
ASR recognized strings	10.42%	9.12%	12%

Table 1: Classification error rate before and after discriminative training using stop word filtering and 756 features, consisting of 62 trigrams, 274 bigrams, 420 unigrams.

What happens if we use less detailed features in the classifier? As an example, the trigram and bigram features are excluded, leaving only 420 unigram features in the routing matrix. As can be expected, the classification results are worse, the error rate increasing from 7.8% to 12.7%. Using discriminative training on the unigram features, however, the error rate can be reduced to 5.5%, representing a relative improvement of about 56%, and identical to the best error rate with trigram, bigram, and unigram features. Using ASR recognized strings, a relative improvement of 44% is achieved, and the resulting error rate is below 10%.

	baseline	after DT	%change
human transcription	12.7%	5.54%	56%
ASR recognized strings	16.9%	9.45%	44%

Table 2: Classification error rate before and after discriminative training using stop word filtering and 420 unigram features.

Recall that the original routing matrix was constructed from n-gram features after preprocessing where a list of ignore words are eliminated and a list of stop words replaced with place holders. The lists of stop and ignore words are typically manually constructed, because the words which may be considered semantically unimportant are different for different contexts and applications. Because such lists require some degree of linguistic knowledge and hand-tuning, supporting natural language call routers for many different domains can be very difficult if not impossible. It is therefore desirable to fully automate the call router design so that such human knowledge and hand-tuning are not necessary. Ideally, the router can be trained using only transcribed utterances and routing destination information.

However, the stop and ignore word lists are quite important to filter out words in order to reduce the total number of features and parameters in the routing matrix. As an example, if no stop and ignore words were filtered, there would have been 7434 features (2756 trigrams, 3442 bigrams, and 1236 unigrams) instead of 756 features. The routing matrix would then have about 230 times more parameters (7434×23 in total).

Since we saw earlier that unigram features together with discriminative training achieved similar results as with all features, we now choose to use only the unigram features (1236 in total) from the list of features which had not been processed with stop word filtering. By not using stop word filtering, we can fully automate the training since no human knowledge is now required to construct the stop word list.

Table 3 shows the results of using these 1236 unigram features without stop word filtering. The baseline result (9.1%) is worse than that using all features, as expected and comparable (slightly better) than the results from using the unigram features which had undergone stop word filtering. Nevertheless, we see that after discriminative training, the error rate again has been reduced significantly (by 39%) to 5.5%, a result similar to the other two feature sets. We see therefore that discriminative training is able to bring all three feature sets to the same classification results. For ASR recognized strings, again the final result after discriminative training is similar to the other two cases.

	baseline	after DT	%change
human transcription	9.1%	5.54%	39%
ASR recognized strings	12.7%	8.79%	31%

Table 3: Classification error rate before and after discriminative training without stop word filtering (fully automatic) and 1236 unigram features.

We are thus able to achieve similar classification results as the

human optimized system using only unigram features which are automatically extracted from the training data without any human expert intervention or specification of stop word lists.

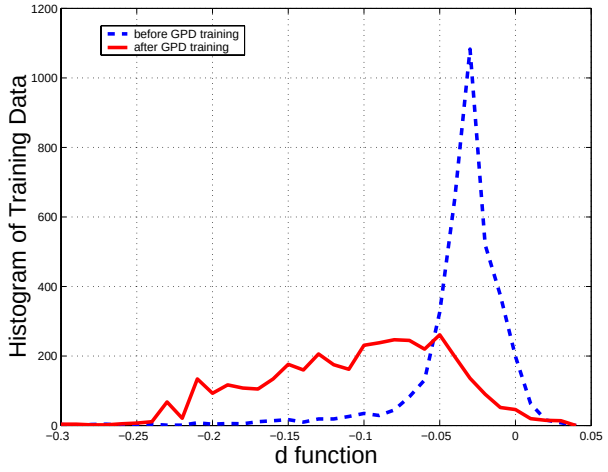

Figure 1: Distribution of the misclassification function is shifted left after discriminative training, yielding a more robust classifier.

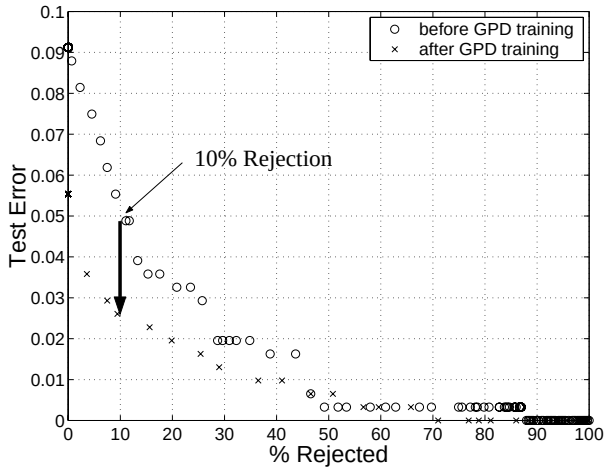

Figure 2: Error rate versus rejection rate

Discriminative training not only reduces the classification error rate, but also improves the robustness of the classifier. Figure 1 shows the distribution of the misclassification function (as described in Equation 3) of the training data before and after discriminative training. The distribution has been shifted left, indicating a more robust classifier. (Recall that a positive value implies misclassification.)

Figure 2 shows the classification error rate of the test data when a subset is rejected using the difference in scores of the top two candidates. At 0% rejection, as seen earlier, the error rate is reduced from 9.1% to 5.5% after discriminative training. At 10% utterance rejection for the test data, there is a relative error reduction of about 40% (from 5% to 3%). At almost all levels of rejection, the discriminative training consistently does better than the baseline because the separation of the correct class from the competing classes has been increased. Therefore, the utility of discriminative training is not only in reducing the classification error rate, but also in improving the robustness of the classifier.

In the original routing matrix, all the elements are positive because they were derived from the counts of the occurrence of the terms. The discriminative training procedure, as we have formulated it, does not guarantee that the parameters remain positive. In fact, checking the routing matrix after the training reveals that many of the elements have now become negative. This makes sense intuitively since the presence of some terms can provide *negative* evidence against a particular destination, particularly when they are helpful in distinguishing a class from its closest competitors.

7. CONCLUSIONS

In this paper, we achieved the goal of fully automating the training of the routing matrix while still maintaining the same level of performance (over 90% accuracy) as that in an optimized system. Specifically, the need for human design of stop and ignore words was eliminated, so that call routers can be trained using just the transcriptions of routed calls.

Eliminating stop word filtering increases the number of features, and this increase was compensated by using only single word terms and eliminating the word pairs and triplets. We showed that the resulting performance degradation can be compensated entirely by minimum error classification (MCE) training. The discriminative training also increased the robustness of the classifier.

We believe the proposed techniques are applicable to algorithms addressing a broad range of speech understanding, topic identification, and information retrieval problems. However, more testing needs to be done to see if they are indeed applicable to many different domains, and as an important example, if they can be used in applications such as internet search engines.

8. ACKNOWLEDGEMENTS

We are grateful for Dr. Jennifer Chu-Carroll's help with understanding the baseline natural language call routing system and sharing the data and code. We also appeciate conversations with Dr. Frank Soong and his ideas.

9. REFERENCES

[1] B. Carpenter and J. Chu-Carroll, "Natural language call routing: A robust, self-organizing approach," in *ICSLP-98*, (Sydney, Australia), pp. 2059–2062, Dec. 1998.

[2] J. Chu-Carroll and B. Carpenter, "Dialogue management in vector-based call routing," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL98)*, pp. 256–262, 1999.

[3] J. Chu-Carroll and B. Carpenter, "Vector-based natural language call routing," *Computational Linguistics*, vol. 25, no. 3, pp. 361–388, 1999.

[4] A. L. Gorin, "Processing of semantic information in fluently spoken language," in *ICSLP-96*, (Philadelphia, PA), pp. 1001–1004, 1996.

[5] H.-K. J. Kuo and C.-H. Lee, "Discriminative training in natural language call routing," in *ICSLP-2000*, (Beijing, China), Oct. 2000.

[6] C.-H. Lee, B. Carpenter, W. Chou, J. Chu-Carroll, W. Reichl, A. Saad, and Q. Zhou, "A study on natural language call routing," in *Proceedings of the Interactive Voice Technology for Telecommunication Applications Workshop*, 1998.