A Portability Study on Natural Language Call Steering

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

Abstract

In this paper we examine the portability of the vector-based call router to a new task involving calls to the operator in the UK. One component of the router was shown to require expert knowledge and hand-tuning: the stop word list. Stop word filtering involves replacing certain words with place markers and is necessary to reduce the number of features and parameters used by the classifier. Two specific approaches that eliminates the need for stop word filtering were investigated that led to comparable classification performance: (1) using trigram, bigram, and unigram features and using SVD to reduce the number of parameters, and (2) using only unigram features and applying discriminative training to boost the performance. After discriminative training, the classification error rate was reduced by 18-30% over the baseline unigram results. Increased robustness is demonstrated by a 24-48% reduction in error rate at 20% false rejection rate.

1. Introduction

Touch-tone menus are very popular for steering callers to different departments in a call center, but they are often frustrating to callers because the list of options may be long and what they need may not even appear to be related to the given choices. In natural language call steering (or routing), the caller is given a chance to say what he/she wants and is automatically steered to the right department or directed to a human operator if the machine is unable to determine the caller’s intent with certainty. The goal is to achieve some level of automation without sacrificing customer satisfaction.

Portability in the design of natural language call steering is one of the most important issues. The successful deployment of such systems can be limited by the need for costly human intervention to customize systems and optimize performance for each new call center. It is therefore important to find call classification algorithms which are completely portable and require little human intervention and yet are robust to the demands of natural language dialogue.

Natural language dialogue between human and machine is still a challenging research problem despite the apparent success of some commercial telephony applications using speech recognition. Such systems almost always constrain the caller to say a specific set of system-defined keywords that must fit within the finite-state grammar of the system; therefore, user training is necessary to teach the caller what to say. Current systems are appropriate for repeat callers who have successfully adapted to the system but are not appropriate for applications such as customer/operator service, where, although there is a high volume of calls, each call is typically from a unique person who may use the service only once and therefore would not appreciate taking time to learn how to interact with the machine. Experienced callers may learn “machine talk” by saying key phrases to which the machine has previously shown understanding, but first-time callers may not even know they are talking to a machine. The caller cannot be prompted with a long list of options, otherwise the service will be as cumbersome as long touch-tone menus.

The caller typically knows what they want to do but often does not know the correct department name. 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 hand-designed finite-state grammar, because what the callers may say cannot be fully anticipated. Instead, requests from real callers have to be collected to train probabilistic language models for the system. Data-driven techniques are therefore essential in the design of such systems. Because of disfluencies and misrecognition by the automatic speech recognizer (ASR), a robust natural language call router/classifier is also necessary in order to process the recognized speech and determine where the call should be steered or whether the system should initiate a disambiguation dialogue.

Many papers have been published in recent years on natural language call steering. Among the approaches are those using a vector-based information retrieval technique [4, 1, 3, 10] and using a probabilistic model with salient phrases [7, 11]. Discriminative training was shown to improve both the classification rate and the robustness of the classifier [8] and to enable a simpler design [9] with no loss in performance by eliminating the need for stop word filtering which requires expert hand-tuning.

Recent papers [2, 5] have described results of joint collaborative research between BTexACT and Lucent Bell Labs on natural language call steering in the UK, including real field trials. The OASIS task involves calls to operator assistance, accessed through the well known “100” code in the UK [6], where people call for many purposes, including problems with the phone line, setting up alarm calls, directory enquiries, money lost in the pay-phone, etc. This task is very challenging: in human-human recordings, the caller can be very conversational, saying over 300 words in a single utterance.

In this paper we examine portability issues when designing a new call classifier for the OASIS task, based on our experience with the USAA (banking) call routing task. In particular, we show that stop word lists are often unreliable when transported to a new domain and thus require costly hand-tuning and expert knowledge. Instead, we follow a fully data-driven approach with no hand-tuning. We show two viable approaches to dealing with the resulting large number of parameters when no stop words are filtered. One method is to use SVD as proposed in [1]. Another approach is to discriminatively train a classifier using only unigram (single word) features [8, 9]. We show that discriminative training reduced the error rate by about 18-30%. Discriminative training also increased the robustness of the classifier such that at 20% false rejection, there was a relative error rate reduction of about 24-48%.
2. Vector-Based Natural Language Call Steering

In vector-based natural language call steering, call steering 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 [10, 1, 3, 4].

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. (Note that this terminology should not be confused with how these terms are used for the n-gram probabilistic language model, where n-gram refers to a model of the probability of a word given its history. Strictly speaking, we should be using the terms single words, word pairs and word triplets, but to be consistent with previous papers [4], we will use 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 \( r_{uw} \) is the frequency with which term \( u \) occurs in calls to destination \( w \). 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 \( \bar{f}_w \) in the routing matrix \( R \). Let \( \bar{x} \) be the \( m \)-dimensional observation vector representing the weighted terms which have been extracted from the user’s utterance. One possible routing decision is to route to the destination with the highest cosine similarity score:

\[
\text{destination } j = \arg\max_j \cos \theta_j = \arg\max_j \frac{\bar{r}_j \cdot \bar{x}}{||\bar{r}_j|| \cdot ||\bar{x}||}. \tag{1}
\]

Alternatively, the cosine scores can also be transformed by a sigmoid function [4] which transforms the scores into a range that makes it easier to perform rejection.

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. We had recently proposed using discriminative training to optimize the minimum classification error criterion for natural language call routing [8, 9]. 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. The same framework is used in this paper.

3. Experimental setup

Experiments were performed using all of the training and test sets from the OASIS corpus described in [5]. For this paper a version of this database was used which had 15 classes, including an “other” set. The “other” class represents rejected utterances which have to be handled by a human operator. The results reported in [5] used a later version with an additional 10 low-frequency classes added. Classification results have however been found to be broadly comparable between these two different classification schemes.

The first engaged utterance [5] for each call has been transcribed and manually assigned a class label, representing the ground truth of the correct class. Each such utterance is considered a token which can be used for training or testing. There are a total of about 7,400 tokens for training and 1,000 for testing.

The same set of training data was used both to construct the initial routing matrix and for performing discriminative training. 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 simplicity, this paper reports only the results on the held out test set consisting of 1000 calls (human-human dialogue), and not data from the actual trial (human-machine dialogue).

4. Results

The main issue we wanted to investigate was the portability of the natural language call classifier [1, 9] which was designed to route calls in a banking application. To this end, the stop word list previously used for the banking task was considered task-independent and was not modified. A new classifier was automatically built without human intervention.

After appropriate filtering as described previously, the number of features chosen was 1465 (34 trigrams, 276 bigrams, 1155 unigrams). The resulting classification error rates on human transcriptions of the spoken utterances are shown in the first line of Table 1. (Note that the terms “trigram” or “trigram model” are meant to refer to classifier model that uses a feature set containing trigram, bigram, and unigram features.) Using the cosine scores, the classification error was 42.5%, whereas using sigmoid confidence mapping [4], the error rate was 40.4%. Note that the classification error rate in this case is defined to be the errors when classifying to one of the 15 classes. The “other” class is treated just like any of the other 14 classes. We will also report results later in ROC plots that show classification results with rejection.

Also shown in Table 1 are the results when all unigram features were used without any stop word filtering. To reduce the number of features, instances of a certain class such as country names were mapped to the country class. A total of 1526 unigram features were used by the classifier. Surprisingly, the classification accuracy was just as good or even better (by up to 28%) than the classifier using a mixture of trigram, bigram and unigram features. One important reason is that the stop word list was not truly task-independent, leading to important terms being excluded from the feature set. For example, all the words in the sentence “hi can i have the time please” were considered stop words in the banking task. Therefore stop word lists are frequently task dependent and constructing the list can be costly and require expert knowledge and hand-tuning.

Without stop word filtering, the number of features are likely to increase significantly when trigram features are used. For example, without stop word filtering, but using class mapping and frequency pruning, the number of features derived was 14275, consisting of 7000 trigrams, 5749 bigrams, and 1526 unigrams. This represents almost 10 times the number of features were used without any stop word filtering. To reduce the number of features, instances of a certain class such as country names were mapped to the country class. A total of 1526 unigram features were used by the classifier. Surprisingly, the classification accuracy was just as good or even better (by up to 28%) than the classifier using a mixture of trigram, bigram and unigram features. One important reason is that the stop word list was not truly task-independent, leading to important terms being excluded from the feature set. For example, all the words in the sentence “hi can i have the time please” were considered stop words in the banking task. Therefore stop word lists are frequently task dependent and constructing the list can be costly and require expert knowledge and hand-tuning.

Table 2 shows the resulting classification error rates, includ-
Table 1: Classification error rates of transcribed utterances using classifiers having different sets of features. The first set consists of 34 trigram, 276 bigram, and 1155 unigram features derived after stop word filtering. The second set was derived without stop word filtering and consists of only unigram features.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Cosine Scores</th>
<th>Sigmoid Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigram + task indep. stop words</td>
<td>42.5%</td>
<td>40.4%</td>
</tr>
<tr>
<td>Unigram, no stop words</td>
<td>40.6%</td>
<td>29.5%</td>
</tr>
<tr>
<td>%difference</td>
<td>4.5%</td>
<td>28%</td>
</tr>
</tbody>
</table>

Table 2: Classification error rate for the classifier using 7000 trigram, 5749 bigram, and 1526 unigram features, as compared with the baseline unigram model (1526 unigram features).

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Cosine Scores</th>
<th>Sigmoid Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human transcription</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unigram, no stop words</td>
<td>40.6%</td>
<td>29.3%</td>
</tr>
<tr>
<td>Trigram, no stop words</td>
<td>23.1%</td>
<td>22.7%</td>
</tr>
<tr>
<td>%difference</td>
<td>43%</td>
<td>23%</td>
</tr>
<tr>
<td>ASR recognized strings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unigram, no stop words</td>
<td>48.7%</td>
<td>41.9%</td>
</tr>
<tr>
<td>Trigram, no stop words</td>
<td>37.2%</td>
<td>34.7%</td>
</tr>
<tr>
<td>%difference</td>
<td>24%</td>
<td>17%</td>
</tr>
</tbody>
</table>

Table 3: Classification error rate before and after discriminative training using 1526 unigram features.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Cosine Scores</th>
<th>Sigmoid Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human transcription</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unigram, no stop words</td>
<td>40.6%</td>
<td>29.3%</td>
</tr>
<tr>
<td>After DT</td>
<td>21.1%</td>
<td>21.9%</td>
</tr>
<tr>
<td>%change</td>
<td>48%</td>
<td>20%</td>
</tr>
<tr>
<td>ASR recognized strings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unigram, no stop words</td>
<td>48.7%</td>
<td>41.9%</td>
</tr>
<tr>
<td>After DT</td>
<td>33.9%</td>
<td>34.5%</td>
</tr>
<tr>
<td>%change</td>
<td>30%</td>
<td>18%</td>
</tr>
</tbody>
</table>

Our recent paper [9] showed that the gap in performance between using the trigram and unigram models in the USAA call routing task can be reduced or even eliminated by using discriminative training, and we wanted to see if this conclusion is also valid for the OASIS task. Table 3 shows the results before and after applying discriminative training on the unigram model. The classification error was indeed brought down to the current best level achieved by the trigram model.

The classification error rates we have mentioned so far treat the “other” class like any of the other 14 classes. However, in order to compensate for the inaccuracy in classification, the call steering system may decide to engage in a disambiguation dialogue or reject the utterance and transfer the call to a human operator [5]. Figure 1 shows the classification accuracy of the test data as a function of the false rejection rate. Tokens are considered rejected if they are either classified into the “other” class or if the confidence score for the classification is low. If the token should not have been rejected (is a member of one of the other 14 classes), this counts as a false rejection. The tokens which were accepted include both those correctly accepted and those falsely accepted. The ratio between the number correctly accepted and the total number accepted is the classification accuracy shown in this plot.

The figure shows the results for three classifiers: the baseline unigram model (represented by circles), the discriminatively trained unigram model (triangles), and the model utilizing trigram, bigram, and unigram features (squares). Also shown are the Rank1 (heavy lines) and Rank2 (lighter lines) curves. Rank1 refers to the classification accuracy where an utterance is considered correctly classified only if the correct class has the highest score. For Rank2, if the correct class has either the best or second best score, the token is considered correctly classified. The curve for Rank2 gives us an idea of how well we may do if the system engages the caller in a disambiguation dialogue by asking them to choose between two destinations. As the figure shows, the Rank2 results are much better than Rank1, especially at low false rejection rates (FRR). At 20% FRR, Rank1 had a classification accuracy of about 75% for the baseline unigram model, but Rank2 was about 87% accurate.

After discriminative training, at 10% false rejection rate, the error rate was reduced from about 28% to 18%, a relative reduction of 36%. At 20% false rejection rate, the error rate was reduced from about 25% to 13%, a relative reduction of 48%. At 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. In fact the Rank1 curve for the discriminatively trained model is very close to the Rank2 curve for the baseline model for FRR over 20%. Therefore, the utility of discriminative training is not only in reducing the classification error rate, but also in improving the robustness of the classifier.

A similar effect is seen in Figure 2 where the input is ASR recognized strings. At 20% FRR, the classification accuracy...
was about 63%. For the discriminatively trained model, the accuracy was about 72%, representing a reduction of 24% in the error rate.

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.

5. Conclusions

In this paper, we addressed the issues of portability for natural language call routers. Based on our experience with the call router for the USAA banking task, we ported our router to the new domain of calls to the operator in the UK, and described the problems encountered.

One main bottleneck in fully automating the training of the call router is the requirement of a list of stop words which we showed was task dependent and inappropriate for the new domain. We successfully demonstrated the feasibility of ideas first introduced in [9] to improve the portability of the natural language call router by fully automating the training of the routing matrix and eliminating the need for specifying a list of stop words. The only human effort required is to supply training material in the form of transcriptions of callers’ utterances and the classes to which they belong.

We achieved good performance for this admittedly difficult task without any tuning and within a few days. (Because the classes in this task are much more confusable than in the USAA call routing task, the absolute classification accuracy is not comparable.) Eliminating stop word filtering greatly increases the number of features, and this increase can be 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. Discriminative training also increased the robustness of the classifier. The discriminatively trained unigram model was shown to have similar performance to the model with trigram, bigram, and unigram features, and had 18-30% fewer errors than the baseline unigram model.

We believe the call steering algorithms that we have proposed are portable, robust, and require no tuning by human experts.

6. Acknowledgements

We are indebted to Mark Farrell for his help with the classification work, especially the nice packaging of the software to compute confusion matrices and ROC plots. Grateful thanks are also extended to the other members of the BT team: David Attwater, Peter Durston, James Allen, Frank Scählill, and others. Thanks to BT Retail who gave permission for the results to be published.

7. References