Semi-automated classifier adaptation for natural language call routing

Silke M. Witt

West Interactive
550 S Winchester Blvd,
San Jose, CA 95128, USA
switt@west.com

Abstract

Commercial spoken dialogue systems traditionally have been static in the sense that once deployed, these applications only get updated as part of formal releases. Also, the creation of classification grammars in natural language call routing applications requires expensive manual annotation of caller intents. The work presented here introduces a process to semi-automatically annotate new data and to use the new annotations to update the training corpus to continually improve the classification performance. This new method consists of a combination of using multiple classifiers in a voting schema to automatically classify an utterance and a boosting mechanism to continually update the classifier with the new automatically annotated training data. This method was tested with 6 weeks' worth of data from a live system. It is shown that with this approach about 94% of all new utterances can be automatically annotated. Using the iterative boosting approach increased the size of the training corpus by about 6% per iteration while at the same time slightly increasing the classification accuracy.

Index Terms: spoken dialog systems, natural language call routing, classifier, semi-automated annotation, multi-classifier voting algorithm

1. Introduction

Commercial spoken dialog systems often handle thousands if not tens of thousands of calls a weekly and yet only a tiny fraction of the data from those calls is analyzed and utilized in order to improve the performance of such systems. Typically, these systems are only periodically updated, perhaps once or twice a year based on analysis of manual transcriptions and grammar coverage tests.

A common type of commercial spoken dialog systems are call routing or *"How may I help you?"* applications that recognize caller utterances such as *'I'd like to check my balance*" or "*I'm missing a channel*". These utterances are first recognized with a statistical language model and then the resulting text string is fed to a classifier. For a more detailed description of such systems, see [1].

To update the language model as well as the classifier of such systems, requires large amounts of manual labor that is very expensive and time-consuming and thus tends to only be done when absolutely necessary. First, the caller utterances are manually transcribed. Secondly, the transcribed utterances are annotated with the desired target classification, where, 'annotation' denotes the process of assigning one of a fixed number of predefined call types to each utterance.

This situation motivated us to investigate methods to both reduce the amount of manual labor required to update a classifier and to utilize available, live data on a regular basis.

The work presented here, focuses on a semi-automated approach to adapt a classifier at regular intervals, such as weekly, in order to both improve classification performance over time and even more importantly, to be able to maintain performance levels when changes in caller patterns occur.

The following section describes related work in more detail. In section 3 we describe the classifiers used and the proposed iterative boosting algorithm. Section 4 presents experimental results based on live, production data. The paper concludes with a discussion of the results and future work.

2. Related work

Semi-supervised learning algorithms that utilize both labeled and unlabeled data have been used now for a while in text classification work in order to minimize the need to label documents by hand. For a summary see [2].

For the classification task in natural language call routing applications, a wide range of different statistical classifiers have been proposed in recent years in order to maximize the classification accuracy and thus the call routing accuracy, see for example [2,3]. Common classifier types are decision tree classifiers, NaïveBayes classifiers, Expectation Maximization and Maximum Entropic classifiers. The performance of such classifiers typically depends on the amount of training data available,

Additionally, there has been considerable effort in the research community in the area of unsupervised and/or semi-supervised adaption of both language models and call category classifiers. In [4], the authors propose a boosting algorithm to improve a classifier iteratively in order to minimize the training error. In [5]. Sarikaya et al., report on using multiple classifiers (Support Vector Machine, Maximum Entropy and NaïveBayes) to automatically annotate unlabeled data. When all classifiers agreed on a label these utterances were added to the training corpus. Wang. et al., see [5] reports classification rate improvements with a classification model that solely utilizes ASR recognition results as input to the training set compared to small amounts of transcribed utterances. Lastly, [7] describes a bootstrapping methodology for semi-automatic annotation using support vector machines.

The work presented here, builds on variations of all this preceding work, with the main difference that we exploit the NBest confidence information of the main production classifier to guide the automated reclassification with additional classifiers. Moreover, we describe an iterative classifier update process that can be implemented in a

commercial production environment in order to both increase coverage and accuracy of a classifier over time. More importantly this method helps to ensure that the performance of a classifier does not decrease over time due to a mismatch of live and training data, since call reasons and their frequency tend to change over time.

3. Automatic classification with a multiple classifiers

The purpose of utterance classification in a call routing system is to assign a confidence score or binary value to each pair $\langle F, c_i \rangle \in \mathcal{F} \times \mathcal{C}$, where $\mathcal{F}$ is the set of features extracted from the spoken utterance u, and $\mathcal{C}$ is a predefined set of call categories.

As summarized in the previous section, several authors have shown that the use of multiple classifiers can improve the overall classifier performance since different classifiers have different strengths. [2] proposes three different methods of classifier combination. The first combines the classifier outputs in a weighted combination. The second to applies a top learner, like a decision tree, to decide on the call category. In the third, a cascading approach is proposed.

The work presented here, proposes a combination of a multi-classifier approach with the NBest results list of a primary classifier. This approach was chosen based on the fact that if newly transcribed utterances are being classified with the existing production classifier, the correct annotation is in the top 3 NBest results for 96% of all utterances.

Typically, classification accuracy is very high for common phrases that occur over and over, whereas classification errors tend to occur for phrases with unknown vocabulary or rarely used ways of describing call reasons. Thus, the approach presented here focuses on improving the automatic classification of low confidence utterances, since the goal of this work is to both increase the **accuracy** as well as the **coverage** of this classifier.

The next three subsections describe the three classifiers that have been used in the follow on experiments.

3.1. Production Classifier

The data and the main classifier that has been used for this work are part of a commercial call routing application in the cable-television domain. The application utilizes a decision tree based classifier that comes with the Nuance OSR recognition engine, [9].

This classifier has been trained on 10,420 unique caller utterances that had been manually transcribed and then manually annotated with one of 45 categories. This amount of training data resulted in a classifier with a baseline classification accuracy of 85.3%. For each utterance classification, the classifier returns an NBest list of 3 results together with their respective confidence.

.

3.2. NaïveBayes Classifier

A common, basic classifier used for caller intent classification is a NaïveBayes classifier. This classifier is based on the NaïveBayes algorithm which uses the Bayes rule to express $P(c_i|F)$ in terms of $P(c_i)$ and $P(F|c_i)$:

$$P(c_i|F) = \frac{P(c_i)P(F|c_i)}{P(F|c_i)} \quad (1)$$

This classifier is called 'naïve' because of the 'naïve' assumption that all features are independent, given the category. We used the Python NL Toolkit, [10] to train a NaïveBayes Classifier with unigram, bigram and trigram features. Before feeding the training data to the classifier, all utterances were stemmed using the WordNetLemmatizer, see also [10]. The accuracy of this classifier on the testset was 85%.

3.3. DecisionTree Classifier

A decisionTree classifier is also used. A decision tree classifier is a classifier model that in order to determine which category to assign to a caller utterance, it utilizes a tree structure where the branches correspond to conditions on feature values, and leaves correspond to category assignments. A classification starts at the initial decision node, known as its root node. Each node contains a condition that checks the match of the test utterances against the feature set, see [10] for more details.

Experiments with different feature sets, showed the best performance was for a feature that consisted of a combination of unigrams, bigrams and trigrams. The accuracy of this classifier on the testset was 86%.

3.4. Multi-Classifier Reclassification

Just like in [5], the auto-classification algorithm presented here is based on the assumption that different classifiers will make different kind of errors. From the production classifier 3 NBest classification results are available. Assuming a confidence threshold of 0.6, the classification error rate (CER) of the production classifier becomes very small, <2%.

Thus, high confidence classification results are being assumed to be correct and not further reclassified. Low confidence utterances on the other had are being reclassified based on additional information from the NaïveBayes and Decision Tree classifiers.

Let $C_{Reclass}(u_i)$ be the reclassification of utterance u_i, then the reclassification rule can be described as:

$$C_{Reclass}(u_i) = \begin{cases} C_{\text{Tree}}(u_i) & \text{if } \begin{cases} C_{\text{Tree}}(u_i) = C_{\text{Bayes}}(u_i) \\ \text{AND} \\ C_{\text{Tree}}(u_i) = C_{\text{nbest1,2 or 3}}(u_i) \end{cases} \\ \text{reject classification} & \text{otherwise} \end{cases} \quad (2)$$

In other words, the multiple classifier voting rule is that for low confidence classification results, if both NaïveBayes and the DecisionTree result in the same classification and if this classification matches one of the 3 NBest production classifications, then assign this classification to utterance u_i. This reclassification process is entirely automated, but for now assumes manual transcription of utterances with low confidence recognition results in order to avoid recognition error propagation, see the next section for more details.

3.5. Iterative classifier update process

When a classifier is being created, it is only as good as the training data available. Ideally, a classifier would continually learn from often thousands of daily calls such systems tend to receive. In order to put a continuous learning and update process in place, the main requirement is to find a way to add new utterances and associated classifications into the classifier with minimal human intervention.

Figure 1 below shows the proposed algorithm of an iterative, automated classifier update cycle. The only component that requires manual intervention is the transcription process of low ASR confidence utterances.

Assuming a regular update interval of a few days or weeks, the steps of the process are:

1. Transcribe all rejected, low confidence or disconfirmed utterances for the given time interval
2. Merge the manual data with high confidence ASR recognition utterances.
3. Classify all utterances with the production classifier using a *NBest=3*.
4. Reclassify low confidence classification results with the NaiveBayes and DecisionTree classification results
5. Update production classifier with auto-classified utterances.
6. Re-start process

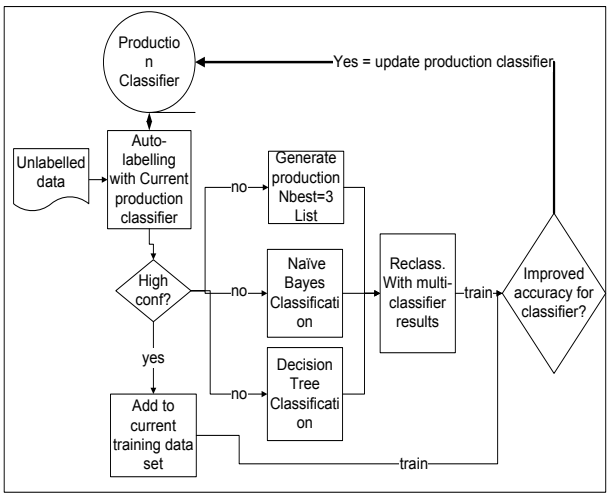

Figure 1: *Schematic diagram of the semi-automated classifier update process.*

The next section describes the implementation of this algorithm and the improvements on both coverage and classification accuracy that were achieved.

4. Experimental Results

Given the production classifier described in section 3.1, a total of 24,013 utterances had been collected over a six week time period. The data was broken up into six data segments. The last data set has been held out from the training data and used as a test set for this experiment.

The goal was to optimize the production classifier with minimal effort on both the transcription and particularly the annotation part.

While the production classifier had very high accuracy on common utterances such 'I have no cable', both the recognition accuracy and the classification of longer utterances was low. As described in section 3.5, only utterances with a low recognition confidence or utterances where the system's classification was disconfirmed by a caller in the follow-on confirmation state, were manually transcribed. Applying these rules amounted to only transcribing about 35% of the total set of utterances. See Table 1 for the breakdown between manually transcribed utterances and ASR based transcriptions.

Table 1 also shows that an initial auto-classification of about 12.4% on average can be achieved by either finding an exact match of a (stemmed) utterance in the training set or by having a high confidence classification with the baseline classifier.

Data set name	# manual trans	# ASR trans	% Unclassified per baseline
Set 1	1318	2937	11.95%
Set 2	1731	4508	11.37%
Set 3	1661	5669	10.54%
Set 4	1331	4317	9.40%
Set 5	988	3511	19.17%
Test Set	1212	3071	15.2%
Total	**8241**	**24013**	**12.4%**

Table 1: Available data sets of unlabeled data

Note, that because the focus of this work was on broadening and expanding the coverage of the language model and the classifier, only 50% of the high confidence utterances were combined with the manually transcribed low confidence utterances. That way, a bias towards less common utterances was introduced.

Given these datasets, the iterative auto-classification and classifier update process was applied to the 5 datasets and the performance was measured against the hold-out test set.

Iteration	Training set size	% Training set size increase	new vocab for adding set 7	% vocab coverage increase
1	39801	0%	116	0.0%
2	41418	4.1%	107	7.8%
3	44720	12.4%	105	9.5%
4	48604	22.1%	100	13.8%
5	51671	29.8%	96	17.2%

Table 2: Training set and vocabulary coverage increase due to iterative auto-classification

Table 2, shows the results of this process with regard to coverage. The coverage increase is described by both the overall increase in the training set as well as the increase in the

vocabulary. As can be seen, with each iteration a small set of new vocabulary is added with the result that in the future a higher percentage of the vocabulary is known.

Next, Table 3 shows the impact of the iterative multi-classifier boosting approach on the classification percentage as well as the classification error rate. On average, the multi-classifier reclassification results in a 30% reduction of the low-confidence classification results of the production classifier for each iteration (see column 3 of Table 3). This translates to an absolute classification rate increase of about 5%. Table 3 also shows that with the boosting approach, on average, 94% of all utterances can be classified automatically, leaving only a handful of utterances for manual review.

Iteration	% classified with high conf. or exact utt match	% reduction in unclass. utts	Total % of automatic classification
1	88.05%	32.1%	93.32%
2	88.63%	29.0%	93.35%
3	89.46%	33.5%	94.08%
4	90.60%	36.2%	94.44%
5	80.83%	22.6%	95.26%

Table 3: Classification coverage increase over time

Last but not least, Figure 2 shows the increase in classification accuracy with each iteration. The data from both Table 3 and Figure 2 show that the added coverage to the training set doesn't come at the cost of a reduction in accuracy.

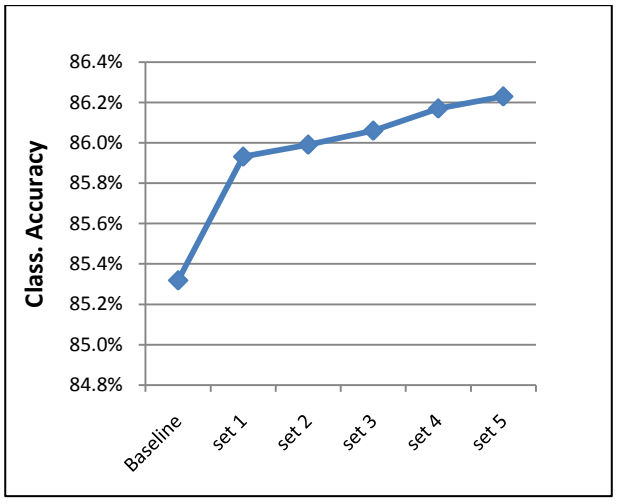

Figure 2: Classification accuracy increase with more iterations of an updated classifier with auto-classified data.

5. Conclusions and Future Work

This paper describes an algorithm to continually update and improve the classification performance of a natural language call routing application where the manual effort is limited to transcribing utterances with low ASR confidence. It was shown that with this approach more than 93% of all utterances can be classified and used to update a production classifier on an ongoing basis.

Moreover, it was shown that the classification rate increases with each iteration. These data have shown that the proposed algorithm is powerful both in terms of increasing coverage as well as accuracy.

Future work will focus on increasing the auto-classification rate even more by either adding additional classifier types such as maximum entropy and expectation maximization classifiers. Additionally, we plan to experiment with sub classifiers that are only trained on the categories that are listed in the NBest lists. Lastly, we will work on reducing the manual transcription effort by sending the low confidence utterance through an offline large-vocabulary recognizer and/or using methods based on cross-valuation as described in [6].

6. References

[1] Gorin, A.L., Riccardi, G. and Wright, J.H., "Automated natural spoken dialog, " IEEE Computer Magazine, vol 35, no. 4, pp. 51-56, April 2002.

[2] Karahan, M., Hakkani-Tur, D., Riccardi, G., Tur, G., "Combining classifiers for spoken language understanding", Proceedings of the ASRU 2003, Merano, Italy, 2003.

[3] Liu, P., Jiang, H., and Zitouni, I., "Discriminative Training of Naïve Bayes Classifiers for natural language call routing", Interspeech 2004, Jeju Island, Korea, 2004.

[4] Tur, G. and Hakkani-Tur, D., "Exploiting unlabeled utterances for spoken language understanding", Eurospeech 2003, Geneva, Switzerland.

[5] Sarikaya,R.,Kuo, J. H.-K., Goel, V. and Gao, Y., "Exploiting unlabeled data using multiple classifiers for improved natural language call-routing", Proceedings of Interspeech 2005, Lisboa, Portugal, 2005.

[6] Wang Y.-Y., Lee, J. and Acero, A., "Speech utterances classification model training without manual transcriptions", Proceedings of the ICASSP, Toulouse, France, 2006

[7] Sarikaya, R., Gao, Y, and Virga, P., "Fast Semi-automatic semantic annotation for spoken dialog systems. Proceedings of the CSLP-2004, Jeju Island, South Korea, 2004.

[8] Tur, G., Schapire, R.E. and Hakkani-Tur,D., "Active Learning for Spoken Language Understanding", Proceedings of the ICASSP-2003, Hong Kong, 2003.

[9] Nuance OSR Developer Guide, Nuance Inc., 2008

[10] Bird, S., Klein, E. and Loper, E., "Natural Language Processing with Python --- Analyzing Text with the Natural Language Toolkit", O'Reilly, 2009