Semantic Data Mining of Short Utterances

Lee Begeja[1], *Harris Drucker*[2], *David Gibbon*[1], *Patrick Haffner*[1], *Zhu Liu*[1], *Bernard Renger*[1], *Behzad Shahraray*[1]

[1]AT&T Labs Research, IP & Voice Services Lab, NJ, USA
[2]Monmouth University, West Long Branch, NJ, USA

***Abstract*— This paper introduces a methodology for speech data mining along with the tools that the methodology requires. We show how they increase the productivity of the analyst who seeks relationships among the contents of multiple utterances and ultimately must link some newly discovered context into testable hypotheses about new information.**

While in its simplest form one can extend text data mining to speech data mining by using text tools on the output of a speech recognizer, we have found that it is not optimal. We show how data mining techniques that are typically applied to text should be modified to enable an analyst to do effective semantic data mining on a large collection of short speech utterances.

***Index Terms*—classifiers, clustering, data reduction, relevance feedback, speech data mining**

I. Introduction

Our work on semantic data mining of short utterances relates to the design of a taxonomy that covers the initial set of utterances, with a specific set of utterance types. This taxonomy relates to a specific business problem of interest to the analyst, who is a subject matter expert in this specific business area. An effective taxonomy will be a set of utterance types such that this set of types covers the preponderance of the utterances in the utterance set. As an example, the utterance, "I wanna order a calling card for my business line," would be mapped to the utterance type, Request(Order_CallingCard). Utterances may have multiple types. The set of utterance types forms the taxonomy of interest and each utterance type is a testable hypothesis when expressed as an NLU classifier.

The overall goal is to develop an effective dialogue response system for use in large scale telephony applications We begin by collecting thousands of utterances in order to effectively cover the space. Initial data collection is done through a "wizard" system that collects the set of utterances in the context of the specific business problem [1]. Once collected, the analyst classifies the utterances and develops a labeling guide that documents the taxonomy. This taxonomy forms the basis for a set of Natural Language Understanding classifiers, which have a one-to-one relationship with the set of utterance types. At this point a separate group of people, called labelers, use the labeling guide as the basis to classify a larger set of utterances. Once the utterances are classified they serve as input to build the NLU classifiers. The ultimate goal would be an effective set of NLU classifiers that could be used with a dialogue manager that will understand and properly reply to people calling in to a telephone voice response unit [2].

We test the NLU classifiers in the field to determine their effectiveness in combination with the dialogue manager. In many instances this combination may not completely satisfy the business problem. This initiates an interactive process that often requires an adjustment to the taxonomy.

In this paper will we show how and why we adapted the following techniques to work on short utterances:

- Data Reduction
- Clustering
- Relevance Feedback

In addition we produce an NLU metric that gives a measure of accuracy for the coverage of the taxonomy. Using this metric an analyst can refine the taxonomy before it goes to the labelers and especially before it goes to the field.

II. Data Reduction

After data collection, the utterances or documents are mapped into a feature vector space for subsequent processing. For many speech data collections, utterance redundancy (and even repetition) is inherent in the collection process and this is tedious for analysts to deal with as they examine and work with the dataset. Natural language processing techniques including text normalization, named entity extraction, and feature computation are used to coalesce similar documents.

A. Text Normalization

In data reduction, we must carefully define what is meant when we say that utterances are "similar". There is no doubt that the user interface does not need to display exact text duplicates. At the next level, utterances may differ only by transcription variants like "100" vs. "one hundred" or of transcription markup such as: "uh, eh, background noise."." *Text normalization* is used to remove this variation.

Text normalization is handled by string replacement mappings using regular expressions. Note that these may be represented as context free grammars and composed with named entity extraction (see below) to perform both operations in a single step. In addition to one-to-one replacements, the normalization includes many-to-one mappings (you ← y'all, ya'll) and many-to-null mappings (to remove noise words).

B. Named Entity Extraction

Utterances that differ only by an entity value should also be collapsed. For example "give me extension 12345" and "give me extension 54321" should be represented by "give me extension *extension_value*." Named entity extraction is implemented through rules encoded using context free grammars in Backus-Naur form.

C. Feature Extraction

To perform processing such as clustering, relevance feedback, or building prototype classifiers, the utterances are represented by feature vectors. At the simplest level, individual words can be used as features. In this case, a lexis or vocabulary for the corpus of utterances is formed and each word is assigned an integer index. Each utterance is then converted to a vector of indices and the subsequent processing operates on these feature vectors. When the dataset available for training is very small (as is the case for relevance feedback) it is best to use less restrictive features to effectively amplify the training data. In this case, we have chosen to use features that are invariant to word position, word count and word morphology and we ignore noise words. With this, the following two utterances have identical feature vector representations:

- I need to check medical claim status
- I need check status of a medical claim

D. Data Reduction Results

The effectiveness of the redundancy removal is largely determined by the nature of the data. As shown in Table I, we have found typical redundancy rates for collections of customer care data of from 30 to 40%. In some cases, where the task is less complex, we have observed data redundancy greater than 50%. Note that as the average length of the documents increases, the redundancy decreases.

Industry Sector	Financial	Health Care	Insurance	Retail
Original Utterances	11,623	12.080	12,109	10,240
Unique Utterances	10,021	10,255	8,865	4956
Unique Utterances after Text Normalization	9,670	9,452	8,103	4,392
Unique Utterances after Entity Extraction	9,165	9,382	7,963	4,318
Unique Utterances after Feature Extraction	7,929	7,946	6,530	3,566
Redundancy	31.8%	34.2%	46.1%	65.2%

Table I

III. CLUSTERING

While removing redundant data greatly eases the burden on the analyst, we can go a step further by organizing the data into clusters of similar utterances. Unfortunately, available distance metrics for utterance similarity are feature-based and result in lexical clusters rather than clusters of semantically similar utterances. So the goal of this stage of the processing is to add further structure to the collected utterance set so that an analyst can more easily make informed judgments to define the utterance types.

Clustering short utterances is problematic due to the paucity of available lexical features. It is quite common for two utterances to have no common features.

A. Clustering Algorithm

Clustering causes data to be grouped based on intrinsic similarities. In any clustering algorithm, we need to define the similarity (or dissimilarity, which is also called distance) between two samples, and the similarity between two clusters of samples. Specifically, the data samples in our task are short utterances of words. Each utterance is converted into a feature vector, which is an array of terms (words) and their weights. The distance between two utterances is defined as the cosine distance between corresponding feature vectors. Assume **x** and **y** are two feature vectors, the distance d(**x**, **y**) between them is given by

$$d(x, y) = 1 - \frac{x \bullet y}{\|x\| \cdot \|y\|}$$

For all the results presented in this paper, we applied named entity extraction, stop word removal, word stemming, and 1-gram term with binary weights to each utterance to generate the set of feature vectors.

The distance between two clusters is defined as the maximum utterance distance between all pairs of utterances, one from each cluster. Figure 1 illustrates the definition of the cluster distance.

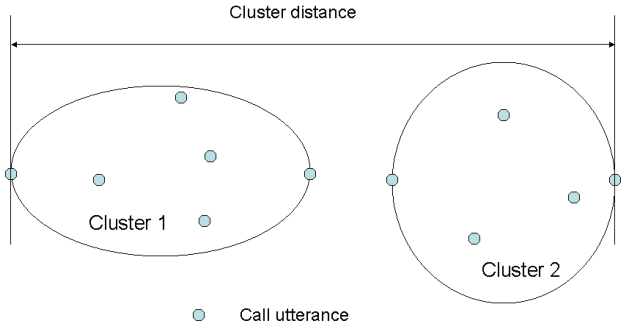

Fig. 1. Illustration of Cluster Distance.

B. Merging Clusters

Clustering may produce a large number of clusters since the utterances are short. To reduce the total number of clusters, we merge all clusters smaller than an established minimum into a special "other" cluster. While there is no set rule for the minimum size of clusters, we find that a minimum of 3 to 5 are reasonable choices in our study.

C. Clustering Performance Evaluation

We use the purity concept explained in [3] to evaluate clustering performance. The two measurements are the average cluster purity (ACP) and the average utterance type purity (ATP), as explained below. First, we define:

n_{ij}: Total number of utterances in cluster i with utterance type j
N_T: Total number of utterance types
N_C: Total number of clusters
N: Total number of utterances
n_j: Total number of utterances with utterance type j
n_i: Total number of utterances in cluster i

The purity of a cluster $p_{i.}$ can then be defined as:

$$p_{i.} = \sum_{j=1}^{N_T} n_{ij}^2 / n_{i.}^2$$

And the average cluster purity (ACP) is:

$$ACP = \frac{1}{N} \sum_{i=1}^{N_C} p_{i.} \cdot n_{i.}$$

Similarly, the utterance type purity $p_{.j}$ and the average utterance type purity (ATP) are calculated as:

$$p_{.j} = \sum_{i=1}^{N_C} n_{ij}^2 / n_{.j}^2$$

$$ATP = \frac{1}{N} \sum_{j=1}^{N_T} p_{.j} \cdot n_{.j}$$

The ATP measures how well the utterances of one utterance type are limited to only one cluster, and the ACP measures how well the utterances in one cluster are within the same utterance type. Two extreme cases are 1) if all utterances are in one cluster, then ATP=100%, and ACP is small; 2) if each utterance is in a separate cluster, then ACP = 100%, and ATP is small. Ideally, we prefer a high ACP and a high ATP for each cluster. When this is not the case (given that the clustering algorithm is used for bootstrapping the utterance types), we prefer a high ACP with reasonable ATP over a high ATP with low ACP(see Table III). In this mode, the analyst does not need to spend too much effort on checking the consistency of each cluster, but rather study the difference and similarity among clusters.

D. Clustering Results

We evaluated the clustering performance on four applications. To generate Table III, we set the clustering distance threshold to 0.6, and the minimum cluster size to 5.

TABLE III
CLUSTERING PERFORMANCE RESULTS BY APPLICATION

Data	N_C	N_T	ATP (%)	ACP(%)
Financial	36	335	23.2	71.2
Health care	92	133	45.6	61.3
Insurance	51	279	25.4	60.2
Retail	31	131	35.8	70.2

IV. Relevance Feedback

Although clustering provides a good starting point, finding all representative utterances belonging to one utterance type is not a trivial task. Additional data mining tools are desirable to help the analyst. Our solution is to provide a classification mechanism based on Support Vector Machine (SVM) classifiers for the analyst to perform this tedious task. In such classification based approaches, the user sequentially assigns labels to examples until the examples belonging to the target utterance type are reasonably separated from the rest. We adopted SVMs as the classifier for two reasons. First, SVMs efficiently handle high dimensional data (in our case, a set of utterances with a large vocabulary). Second, SVMs provide reliable performance with a small amount of training data. Both advantages perfectly match the task at hand. For more details about SVMs, please refer to [4][5].

The most commonly used approach in Information Retrieval (IR) is *relevance feedback*. In essence, an analyst indicates to the retrieval system that it should retrieve "more documents like the ones desired." Selecting relevant documents based on analyst's inputs is basically a classification problem. Relevance feedback is an iterative procedure. The analyst starts with a cluster or a query result by certain keywords, and marks each utterance as either a positive or negative utterance for the utterance type. The analyst's inputs are collected by the relevance feedback engine and they are used to build a SVM classifier that attempts to capture the essence of the utterance type. The SVM classifier is then applied to the rest of the utterances in the dataset and it assigns a relevance score for each utterance. A new set of the most relevant utterances are generated and presented to the analyst, and the second loop of relevance feedback begins.

For efficient labeling of large quantities of data, another iterative approach, generally referred to as *active learning,* is preferred. The most relevant utterances, while interesting from an IR standpoint, are usually obvious for the classifier: they are not those which maximize progress when learning them. It is rather the labeling of uncertain utterances, which lie at the decision boundary, which gives the greatest improvement to the discrimination between relevant and irrelevant utterances. To establish which utterances lay at the decision boundary, one can rely on either geometric or probabilistic criteria.

According to the geometric criterion, the examples which should be labeled in priority stand at the center of the classifier. For an example x, let $g(x)$ be the output of the SVM before the addition of any bias. The geometric criterion relies on the transformation

$$g(x) + b$$

such that for positive support vectors

$$g(x) + b = 1$$

and for negative support vectors

$$g(x) + b = -1$$

The center of the margin corresponds to

$$g(x) + b = 0$$

In our problem we define the positive class as examples belonging to the utterance type and the negative class as all other examples. As a consequence the positive class has many fewer representatives than the negative class. Therefore, choosing examples at the center of the margin will typically return a large majority of negative utterances, and result in a labeling process which is both suboptimal and frustrating.

The probabilistic criterion relies on the fact that the classifier output approximates in a reasonable way the posterior probability that a given utterance belongs to the utterance type and selects examples where the posterior probability is the closest to 0.5. In the case of SVMs, such a probabilistic approximation can be obtained with the application of univariate logistic regression to the output of the SVM [14]. The transformation consists in

$$g'(x) = \sigma(a.g(x) + b)$$

where σ is the sigmoid function. a and b are optimized to minimize the Kullback Liebler divergence between $g'(x)$ and the posterior probability of the class $P(c \mid x)$ given x.

Separate training sets should be used to train the SVM classifier and the logistic parameters a and b. We use cross-validation to maximize the use of labeled examples. Note that in the case of active learning, our logistic remapping function is trained on the already labeled examples, whose distribution is skewed and not statistically representative of the true distribution. Despite this limitation, we found the logistic remapping approximation worked well on unlabeled examples, returning comparable numbers of positive and negative examples, and converging significantly faster than the geometric criterion.

Both theory and computer simulations predict that active learning using the probabilistic criterion minimize the number of examples one has to label to achieve a given classification accuracy on test data. Our simulations suggest that, if the goal is to label enough examples to build a classifier that generalize well on test data, the active learning strategy can reduce by up to a factor of six the number of examples that need to be labeled.

V. NLU METRIC

The analyst can improve utterance types by iteratively building and testing interim NLU classifiers. A Web interface was added to allow the analyst to build and test NLU classifiers and to better understand patterns in the NLU classifier test results. We used BoosTexter as the underlying boosting algorithm for classification [7][8].

After the analyst has labeled the utterances (we will refer to these as truth utterance type labels), approximately 20% of the labeled utterances are set aside for testing. The remaining data are used to build the initial NLU classifier. For each of the tested utterances in the test data, logs show the classification confidence scores for each utterance type. Confidence scores are replaced by probabilities that have been computed using a logistic function. These probabilities are then used to calculate the NLU metric which attempts to reveal patterns in the classification results. The NLU metric, roughly speaking, is a measure of utterance type differentiability. The NLU metric is calculated as follows and is averaged over the utterances that belong to only one utterance type:

$$S = \begin{cases} \frac{1}{N}\sum_{i=1}^{N}(T_i - X_i) & \text{for } T_i = H_i \text{ (correctly classified)} \\ \frac{1}{N}\sum_{i=1}^{N}(T_i - H_i) & \text{for } T_i \neq H_i \text{ (incorrectly classified)} \end{cases}$$

where S is the NLU Metric, N is the number of utterances that belong to only one utterance type, Ti is the truth probability, Xi is the next highest probability, and Hi is the highest probability. A test utterance is correctly classified if the calculated probability of the truth type is the highest probability.

As can be seen in Table IV, the NLU metric for the Request(Order_CC) utterance type is 0.681. Of the 18 test utterances, only two were incorrectly classified. Thus, although some of the test utterances in the test log indicate problems, on the aggregate the NLU metric for this utterance type is quite good. Other good utterance types are shown in Table V. If the NLU metric was less than 0.50 or negative, this would indicate a problem with the utterance type. The best approach for the analyst is to evaluate both the test log probabilities (for utterance level problems) and the NLU metric (for aggregate level problems) for every utterance type.

Utterance type	# of Tests (# correct)	NLU Metric
Report(LostStolen_CC)	31 (30 correct)	0.941
Request(Call_Transfer_CSR)	28 (27 correct)	0.850
Request(Order_CC)	18 (16 correct)	0.681

Table IV. NLU Metric

This metric allows the analyst to identify utterance types that might have problems in the field. Once identified, the analyst could redefine the problematic utterance types. Another interim NLU classifier could then be built and tested to determine if the changes improved the utterance type. The analyst can iteratively build and test the interim NLU classifiers. Once the utterance types are correct the final annotation guide is created. The final annotation guide would then be used by the labelers to label all the utterance data needed to build the final NLU classifier. The NLU metric helps create better utterance types, which ultimately leads to a better NLU classifier.

VI. SUMMARY

We have shown adaptations of text based data mining tools to make them more useful in the context of speech data mining. These tools enable our analysts to develop NLU classifiers in the context of a specific business problem

REFERENCES

[1] A. L. Gorin, G. Riccardi, and J. H. Wright, "How May I Help You?," *Speech Communication*, 1997, 23:113-127

[2] A. Abella and A. Gorin, "Construct algebra: Analytical dialog management," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1999, Washington, D.C., June.

[3] J. Ajmera, H. Bourlard, I. Lapidot and I. McCowan, "Unknown-Multiple Speaker clustering using HMM", ICSLP, Denver, Colorado, 2002, 573-576.

[4] H. Drucker, D. Gibbon, B. Shahraray, "Relevance feedback using support vector machines," in *Proceedings of the 2001 International Conference on Machine Learning*.

[5] V. N. Vapnick, *Statistical Learning Theory*. John Wiley and Sons Inc.,1998.

[6] C. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods", in *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans, eds., pp. 61-74, MIT Press, (1999).

[7] M. Rochery, R. Schapire, M. Rahim, N. Gupta, G. Riccardi, S. Bangalore, H. Alshawi and S. Douglas, "Combining prior knowledge and boosting for call classification in spoken language dialogue," *ICASSP 2002*.

[8] R. Schapire, Y. Singer, 2000. *BoosTexter: A Boosting-based System for Text Categorization*, Machine Learning, 39(2/3):135-168.