FAST SEMI–AUTOMATIC SEMANTIC ANNOTATION FOR SPOKEN DIALOG SYSTEMS
Ruhi Sarikaya†, Yuqing Gao†, and Paola Virga‡
† IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{sarikaya,yuqing}@us.ibm.com
‡ Johns Hopkins University
paola@jhu.edu

ABSTRACT
This paper describes a bootstrapping methodology for semi-automatic semantic annotation of a “mini-corpus” that is conventionally annotated manually to train an initial parser used in natural language understanding (NLU) systems. We propose to cast the problem of semantic annotation as a classification problem: each word is assigned a unique set of semantic tag(s) and/or label(s) from the universal tag/label set. This approach enables “local” semantic annotation resulting in partially annotated sentences. The proposed method reduces the annotation time and cost that forms a major bottleneck in the development of NLU systems. We present a set of experiments conducted on the medical domain “mini-corpus” that contains 10K hand-annotated sentences. Three annotation methods are compared: parser (baseline), similarity and classification-based annotations. The support vector machine (SVM) based classification scheme is shown to outperform both similarity and parsed-based annotation.

1. INTRODUCTION
There are two main approaches to NLU: grammar-based and corpus-driven. The grammar is often handcrafted by a grammarian and/or a domain knowledge expert. As the grammar-rules need to capture domain specific knowledge, pragmatics, syntax and semantics altogether, it is difficult to write a rule-set that has good coverage of real data without becoming intractable [7]. An alternative method is corpus-based approach requiring less developer expertise than traditional grammar-based systems. Corpus-based approaches employ statistical methods to model the syntactic and semantic structure of the sentences. The task of writing grammars is replaced by a simpler task of annotating the meaning of a set of sentences. The corpus-driven approaches are desirable in that the induced grammar can model real data closely. Some grammar induction algorithms can automatically capture patterns in which syntactic structures and semantic categories interleave into a multitude of surface forms. However, manual-annotation of corpora may also be costly.

Collection of a “mini–corpus” of 10–15K sentences is a necessary step for building a conversational system using both corpus-driven and grammar-based methods. Grammar-based methods need the corpus to write and validate the grammar rules.

Our baseline system is a decision tree based statistical parser that is trained using manually annotated data. The parser performance depends heavily on the amount of annotated data. We built NLU applications and trained parsers for several tasks. Based on our experience on these tasks the parser needs at least 10–15K sentences to achieve reasonable performance. Therefore, it is essential to have a “mini–corpus” of about 10K manually-annotated data for any conversational system. The parser trained using the “mini–corpus” can facilitate bootstrapping the annotation of rest of the training data. This would reduce labor in the annotation of the remaining training data since the human annotator would only check and, if needed, correct the automatic annotation.

Thus far, little attention, if any, is given to rapid annotation of the “mini–corpus”. In general, it is assumed that this data is manually-annotated. However, there are cases where one has to deploy a spoken dialog system within a given time frame where speed is the main concern. There are other cases where one has to deploy many systems in different domains. For these scenarios rapid annotation of the “mini–corpus” is crucial. Our work is an attempt to expedite the “mini–corpus” annotation step for both grammar-based and corpus-based NLU. We conceive of several desirable features for such methodology: 1) It should minimize the reliance on the amount of annotated data for reasonable performance, 2) It should be used in both grammar-based and corpus-based NLU frameworks. 3) It should take full advantage of the very limited annotated data (a few thousand sentences). 4) It should be easy to implement.

In this work, we devise a semi-automatic methodology to capture language structures given very limited annotated corpus. Since the “mini–corpus” should be error-free, a completely manual correction step is necessary after semantic analysis (elimination of wrong analysis). We strive to minimize the effort necessary during the manual correction phase of the annotation by word-centric tagging and labelling of the sentences. We formulate the automatic annotation problem as a classification problem. The goal of the annotation is not to fit the best parse tree to a sentence, but to assign the best possible tag and a label to each word to minimize human labor for corrections. The proposed methods possess all of the listed features mentioned above. We use similarity measure and a set of SVM-based classifiers to perform classification. In the classification framework (i.e., similar-
ity and SVM-based classification) each word is guaranteed to be assigned a set of tag and labels. We note that the proposed methods provide a viable alternative to the partial parsing strategy [4] that allows partial interpretations of the sentence segments.

This paper is organized as follows: Section 2 describes the decision tree based statistical parser used in our NLU analysis. Section 3 describes the classification based schemes proposed in this work. We present the experimental results in Section 4 followed by the conclusions in Section 5.

2. THE DECISION TREE BASED STATISTICAL PARSER

Most of the standard statistical parsers are designed based on the premise that there is reasonable amount of manually annotated training data. However, little work, if any, has been done on how these statistical parsers behave in the extreme case when the size of the training data is drastically reduced. During training the parser learns the structure of the sentences. Given a reasonably large amount of training data, there will probably be relatively few “ungrammatical sentences” that the parser fails to generate a complete parse tree. Lack of training data adversely affects parser robustness.

We use the decision tree based statistical parser to extract meaning from an utterance [5]. The objective of the parser is to fit a complete parse tree to a given sentence. The parser works in left–to–right bottom–up fashion. For example, the parser would first attempt to predict the tag for the word “I” in Fig. 1, then predict the label following it, and so on. Each of these decisions corresponds to a parser action. At any given step, the parser performs feature value assignment corresponding to a parser action. Each parser action is assigned a probability given the current context.

For the sake of simplicity, we consider only four main feature values. Let \( N^k = \{ N^k_t, N^k w, N^k l, N^k e \} \) refer to the 4-tuple feature values at the \( k \)th node in the current parse state. These feature values are “label” (\( N^k_t \)), “word” (\( N^k w \)), “tag” (\( N^k l \)), and “extension” (\( N^k e \)). The probability distribution for each feature value is estimated using conditional models. For example, the tag feature is modeled by:

\[
p(N^k_t | context(t)) \approx p(N^k_t | N^{k-1} l, N^{k-2} w, N^{k-3} l, N^{k-4} w, N^{k+1} l, N^{k+2} w)
\]

(1)

The label feature value prediction is also conditioned on a similar set of features.

We recognize that using a large feature set in constraining the prediction of a tag or label has advantages if there is enough training data to learn the dependencies. However, there are many different applications of parsing, and each application has a different cost threshold for efficiency, robustness and accuracy. When the training data is limited, it is likely that the parser does not see some or most of the words in the training data. Therefore, the scores for each parser action falls below a preset threshold. Note that this will happen independent of the threshold value, since each parser action will be “unknown” with respect to training data and the parser

Figure 1: An example of semantically annotated sentence (or a parse tree).

tree generation will fail. For example, if we use only 1K sentences for parser training and parse the next 1K sentences, the parser fails to parse 36% of the sentences since some or most of these sentences contain words that were not covered in the 1K training set. The corresponding rates for 2K, 3K, and 9K sets are 23.5%, 14.7%, and 5.4% respectively. However, note that lack of training data has adverse effects not only for parser but also for the proposed methods.

3. THE CLASSIFICATION–BASED SEMI–AUTOMATIC ANNOTATION SCHEMES

We formulate the problem of semantic annotation as a multi-class classification problem. The first method is based on example–based learning that does not even need training though it needs the training data. The second method builds a classifier in a supervised manner using the limited manually–annotated data. Next, we present the proposed methods in detail.

3.1. SIMILARITY–BASED ANNOTATION

When dealing with limited domains, it is likely that most of the words are used with only one meaning. For example, even though English has several meanings including language, person or discipline, only one of these is likely to be used in a limited domain. However, if one needs to annotate every word in that domain there will be cases where some words may take several meanings and, hence different tag and labels. The method we propose is based on the premise that given two instances of a word, if the context in which they are used is similar, then they should be annotated with the same tag and labels.

The key question is what is the appropriate similarity measure? Inspired by resemblance of the annotation problem and machine translation (MT) evaluation where a translated or a candidate sentence is compared to a set of reference sentences, we adopted BLEU (BiLingual Evaluation Understudy) [3] as the similarity measure for annotation. BLEU is a fully automatic evaluation metric that forms a viable alternative to expensive and time–consuming human judgment of translation quality. The BLEU metric is defined as follows:

\[
\text{BLEU} = \text{BP} \cdot \exp \left( \sum_{n=1}^{N} w_n \log p_n \right)
\]

(2)

where \( N \) is the maximum \( n \)-gram length, \( w_n \) and \( p_n \) are the corresponding weight and precision, respectively, and \( \text{BP} \) is the brevity penalty:

\[
\text{BP} = \begin{cases} 
1 & \text{if } c > r \\
\exp(1 - r/c) & \text{if } c \leq r
\end{cases}
\]

(3)

where \( r \) and \( c \) are the lengths of the reference and candidate sentences, respectively.
Our goal is to annotate words in a sentence rather than to determine how close two sentences are. Therefore, we tailored the way BLEU is applied to our needs. Based on the analogy between MT evaluation and annotation the sentence to be annotated is treated as the “candidate” sentence and all the sentences in the training data containing the word to be annotated are considered as possible “reference” sentences. The task is to find the best reference sentence using the BLEU score. Note that training sentences are used as reference after a processing step where the most relevant segment of the reference sentence is extracted with the purpose of having similar left and right context size of the word to be annotated. Therefore, the reference sentences may be truncated if the context to either side of the word to be annotated are larger than the corresponding sizes of the “candidate” sentence. The annotation is performed sequentially for each word. The best reference sentence is likely to change for each word of a sentence. Once the best reference sentence that contains the word to be annotated is found, the tag and the label of that word is used as the tag and label for the current word. If there is no reference sentence containing the current word then tag and labels are selected based on priors.

3.2. THE MULTI-CLASS CLASSIFICATION FOR ANNOTATION

The semantic annotation problem can be cast as a multi-class classification problem. Any standard machine learning method including Maximum Entropy and Support Vector Machines (SVM) can be used to train a classifier. In this study, we used SVM as the learning method to build a set of classifiers. Although SVM builds binary classifiers, a multi-class classification problem can be performed using pairwise binary classifiers. Namely, one can train \( N(N-1)/2 \) pairwise binary classifiers, where \( N \) is the number of classes. The most important step in this scheme is the relevant feature selection. We derive the features from a context surrounding the word to be annotated. For the sake of simplicity, we assumed an analysis depth of two, where each word is assigned a tag and a label. The classification scheme is sequential. First, the tag of a word is determined using a tag SVM classifier built using the following tag feature vector, \( f_{tag} \) for the \( i \)th word, \( w_i \):

\[
f_{tag} = [w_{i-2}w_{i-1}w_i,w_i+1w_i+2l_{i-2}-tl_{i-1}l_{i+1}l_{i+2}] \quad (4)
\]

where \( w_i \) is the word to be tagged, \( l_{i-1} \) and \( l_{i+1} \) are the tag and label of the previous word, \( l_{i-1} \), respectively. In addition to word context, tags and labels of the previous words are also used. Next, given the predicted tag, \( l_i \), we use the following label feature vector to predict the label for \( w_i \) using a separate label SVM model:

\[
f_{label} = [w_{i-2}w_{i-1}w_i,w_i+1w_i+2l_{i-2}l_{i-1}l_{i+2}l_{i+1}] \quad (5)
\]

Once the label, \( l_i \), for \( w_i \) is determined, then \( l_{i+1} \) and \( l_{i-1} \) are predicted sequentially. In this work, the number of classes for tag and label are 158 and 69, respectively.

Much of the flexibility and classification power of SVM’s resides in the choice of kernel. Some examples are linear, polynomial and radial basis functions. In this work, we chose linear kernels to train the SVM, since the choice of other kernels did not provide improved performance for the current application.

4. EXPERIMENTAL RESULTS

Our experimental corpus is in the medical domain. This mini-corpus is collected for our speech-to-speech translation project [6], where a doctor interacts with a patient. The doctor attempts to understand the patient’s problem and, if needed, prescribes medicine. The corpus has 10k manually annotated sentences. We adopted two scenarios to evaluate parser-based and the proposed annotation schemes. In the first scenario 10k data is uniformly split into 10 equal sets. We used the first 1K as the training data and annotated the second 1K set. Next, the first 2K sentences are used as training data to annotate the third 1K set. The process is repeated until having 9K sentences as training set and annotating the tenth 1K set. In this scenario the test set changes at each step, therefore it is called “incremental” evaluation. The second scenario is called “fixed” evaluation where we fixed the test set and increased the training data incrementally.

Even though the annotation error is our main concern we also provided the \( F \)-measure in Table 1 for all three methods. Precision, recall and \( F \)-measure are widely used for performance evaluation: for our case, recall \((\tau)\) measures the percentage of relevant bracketed annotation generated by a method as compared to total bracketed reference annotation (described in the next page) and precision \((p)\) measures the percentage of relevant bracketed annotation contained in the annotated sentence. The \( F \)-measure is defined using the following formula:

\[
F_\beta = \frac{(1 + \beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}
\]

where \( \beta \) is a weighting factor that determines relative weighting of precision and recall. If \( \beta = 1 \) both precision and recall are equally weighted. If \( \beta = 0 \) then \( F_0 = p \) and as \( \beta \rightarrow \infty, F_\infty \rightarrow r \). For both “incremental” and “fixed” test cases the SVM-based classification scheme has a higher \( F \)-measure than the parser-based method up to 4-5K. They have similar scores for 6-7K and then the parser-based scheme starts to outperform the SVM-based scheme. Similarity-based scheme outperformed the parser-based scheme up to 3K.

The objective of a performance measure should be to minimize the work left to human annotator for the correction of the wrongly annotated examples. Therefore, annotation error rate (AER) that measures the percentage of tags and labels (with respect to total tags and labels in the reference) that needs to be corrected by a human is an appropriate measure. The AER can be approximated by computing the string edit distance between the reference and the hypothesized annotation that are in the bracketed form. For example, the parse given Fig. 1 can be converted to the following bracketed form:

\[[\text{IS! [SUBJECT Lpron-sub SUBJECT] [INTEND need_intend to_intend0 INTEND] [VERB x-ray verb VERB] [BODY-PART your_lpron-pos chest_body-part BODY-PART] IS!}]

Note that when AER is computed [IS! and !IS!] are not included as they always exist in all the sentences. In Fig. 2, we compared the annotation error rates for parser, similarity and SVM-based annotation schemes for the “incremental” test case. We took the parser-based method as the baseline. The similarity-based method provided lower annotation error...
Table 1: Comparison of Annotation Schemes based on F-measure.

<table>
<thead>
<tr>
<th>Test Set</th>
<th>Method</th>
<th>1K</th>
<th>2K</th>
<th>3K</th>
<th>4K</th>
<th>5K</th>
<th>6K</th>
<th>7K</th>
<th>8K</th>
<th>9K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental</td>
<td>Parser</td>
<td>87.3</td>
<td>84.1</td>
<td>81.6</td>
<td>85.1</td>
<td>84.5</td>
<td>75.7</td>
<td>81.6</td>
<td>84.5</td>
<td>75.7</td>
</tr>
<tr>
<td></td>
<td>Similarity</td>
<td>74.5</td>
<td>73.5</td>
<td>71.6</td>
<td>74.5</td>
<td>73.5</td>
<td>64.6</td>
<td>71.6</td>
<td>74.5</td>
<td>64.6</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>74.5</td>
<td>75.6</td>
<td>79.8</td>
<td>81.4</td>
<td>82.1</td>
<td>80.3</td>
<td>84.9</td>
<td>85.5</td>
<td>86.0</td>
</tr>
<tr>
<td>Fixed</td>
<td>Parser</td>
<td>72.6</td>
<td>87.3</td>
<td>88.7</td>
<td>86.7</td>
<td>84.5</td>
<td>72.2</td>
<td>81.6</td>
<td>84.5</td>
<td>72.2</td>
</tr>
<tr>
<td></td>
<td>Similarity</td>
<td>72.2</td>
<td>75.6</td>
<td>79.8</td>
<td>81.4</td>
<td>82.1</td>
<td>80.3</td>
<td>84.9</td>
<td>85.5</td>
<td>86.0</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>74.5</td>
<td>77.4</td>
<td>84.1</td>
<td>86.6</td>
<td>88.8</td>
<td>82.1</td>
<td>85.9</td>
<td>86.9</td>
<td>87.3</td>
</tr>
</tbody>
</table>

Figure 2: Comparison of the annotation schemes for the incremental test data.

Similar observations hold for Fig. 3 where the same methods are compared for the “fixed” test set scenario. Therefore, using the SVM–based classification scheme before the manual annotation of the “mini–corpus” provides significant savings in annotation time and cost.

When we compare the F–measure results in Table 1 and the annotation error rates in Figs. 2 and 3, we notice that the improvements in annotation accuracy are significantly better than the corresponding improvements in F–measure. This is because precision scores are artificially high for the parser–based method since the parser fails to generate complete parse trees for a relatively high percentage of sentences when training data is very small. However, if it generates an output it is likely that it is correct. Therefore, if the data to be annotated is more than mini–corpus size (i.e., ~10K), first using the SVM–based classification method for the incremental annotation and then switching to the parser–based scheme is a reasonable strategy.

5. CONCLUSIONS

We presented two alternative annotation methods for the annotation of the mini–corpus that is used to train the initial parser. Manual annotation of the mini–corpus is one of the major bottlenecks in building natural language understanding systems. The proposed methods treat the annotation as a multi–class classification problem and perform word level “local” annotation of the sentences. The first method is based on a similarity metric and the second method is based on the SVM–based classification. Annotation experiments on the medical domain corpus demonstrated that the similarity–based scheme outperformed the parser–based baseline scheme up to a corpus size of 5K. The SVM–based classification showed significant improvement over the parser–based baseline method in annotation accuracy and reduction in cost and time up to a corpus size of 9K sentences.

Acknowledgment

The authors would like to thank Kishore Papineni, Vaibhava Goel, Jeff Kuo and Bowen Zhou for fruitful discussions.

References