Using Semantic and Syntactic Graphs for Call Classification

Dilek Hakkani-Tür **Gokhan Tur**
AT&T Labs – Research
Florham Park, NJ, 07932
{dtur,gtur}@research.att.com

Ananlada Chotimongkol
Carnegie Mellon University
Pittsburgh, PA 15213
ananlada@cs.cmu.edu

Abstract

In this paper, we introduce a new data representation format for language processing, the syntactic and semantic graphs (SSGs), and show its use for call classification in spoken dialog systems. For each sentence or utterance, these graphs include lexical information (words), syntactic information (such as the part of speech tags of the words and the syntactic parse of the utterance), and semantic information (such as the named entities and semantic role labels). In our experiments, we used written language as the training data while computing SSGs and tested on spoken language. In spite of this mismatch, we have shown that this is a very promising approach for classifying complex examples, and by using SSGs it is possible to reduce the call classification error rate by 4.74% relative.

1 Introduction

Goal-oriented spoken dialog systems aim to identify intents of humans, expressed in natural language, and take actions accordingly to satisfy their requests. The intent of each speaker is identified using a natural language understanding component. This step can be seen as a multi-label, multi-class call classification problem for customer care applications (Gorin et al., 1997; Chu-Carroll and Carpenter, 1999; Gupta et al., To appear, among others). As an example, consider the utterance *I would like to know my account balance*, from a financial domain customer care application. Assuming that the utterance is recognized correctly by the automatic speech recognizer (ASR), the corresponding intent (call-type) would be *Request(Balance)* and the action would be telling the balance to the user after prompting for the account number or routing this call to the billing department.

Typically these application specific call-types are pre-designed and large amounts of utterances manually labeled with call-types are used for training call classification systems. For classification, generally word n-grams are used as features: In the *How May I Help You?*^{SM} (*HMIHY*) call routing system, selected word n-grams, namely *salient phrases*, which are salient to certain call-types play an important role (Gorin et al., 1997). For instance, for the above example, the salient phrase "account balance" is strongly associated with the call-type *Request(Balance)*. Instead of using salient phrases, one can leave the decision of determining useful features (word n-grams) to a classification algorithm used as described in (Di Fabbrizio et al., 2002) and (Gupta et al., To appear). An alternative would be using a vector space model for classification where call-types and utterances are represented as vectors including word n-grams (Chu-Carroll and Carpenter, 1999).

Call classification is similar to text categorization, except the following:

- The utterances are much shorter than typical documents used for text categorization (such as broadcast news or newspaper articles).

Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in NLP, pages 24–31,
Ann Arbor, June 2005.

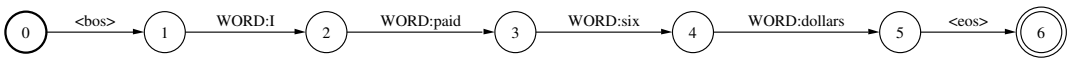

Figure 1: An example utterance represented as a single path FSM.

- Since it deals with spontaneous speech, the utterances frequently include disfluencies or are ungrammatical, and
- ASR output is very noisy, typically one out of every four words is misrecognized (Riccardi and Hakkani-Tür, 2003).

Even though the shortness of the utterances may imply the easiness of the call classification task, unfortunately this is not the case. The call classification error rates typically range between 15% to 30% depending on the application (Gupta et al., To appear). This is mainly due to the data sparseness problem because of the nature of the input. Even for simple call-types like *Request(Balance)*, there are many ways of uttering the same intent. For instance, in one of the applications we used in our experiments, as a response to the greeting prompt, there are 2,697 unique utterances out of 3,547 utterances for that call-type. Some examples include:

- *I would like to know my account balance*
- *How much do I owe you*
- *How much is my bill*
- *What is my current bill*
- *account balance*
- *You can help me by telling me what my phone bill is*
- ...

Given this data sparseness, current classification approaches require an extensive amount of labeled data in order to train a call classification system with a reasonable performance. In this paper, we present methods for extending the classifier's feature set by generalizing word sequences using syntactic and semantic information represented in compact graphs, called syntactic and semantic graphs (SSGs). For each sentence or utterance, these graphs include lexical information (words), syntactic information (such as the part of speech tags of the words and the syntactic parse of the utterance), and semantic information (such as the named entities and semantic role labels). The generalization is expected to help reduce the data sparseness problem by applying various groupings on word sequences. Furthermore, the classifier is provided with additional syntactic and semantic information which might be useful for the call classification task.

In the following section, we describe the syntactic and semantic graphs. In Section 3, we describe our approach for call classification using SSGs. In Section 4, we present the computation of syntactic and semantic information for SSGs. In the last Section, we present our experiments and results using a spoken dialog system AT&T VoiceTone® Spoken Dialog System (Gupta et al., To appear).

2 Semantic and Syntactic Graphs

Consider the typical case, where only lexical information, i.e. word n-grams are used for call classification. This is equivalent to representing the words in an utterance as a directed acyclic graph where the words are the labels of the transitions and then extracting the transition n-grams from it. Figure 1 shows the graph for the example sentence *I paid six dollars*, where <*bos*> and <*eos*> denote the beginning and end of the sentence, respectively.

Syntactic and semantic graphs are also directed acyclic graphs, formed by adding transitions encoding syntactic and semantic categories of words or word sequences to the word graph. The first additional information is the part of speech tags of the words. In the graph, as a parallel transition for each word of the utterance, the part of speech category of the word is added, as shown in Figure 2 for the example sentence. Note that, the word is prefixed by the token *WORD:* and the part-of-speech tag is prefixed by the token *POS:*, in order to distinguish between different types of transitions in the graph.

The other type of information that is encoded in these graphs is the syntactic parse of each utterance, namely the syntactic phrases with their head words. For example in the sentence *I paid six dollars*, *six dollars* is a noun phrase with the head word *dollars*. In Figure 2, the labels of the transitions for syntactic phrases are prefixed by the token *PHRASE:*. There-

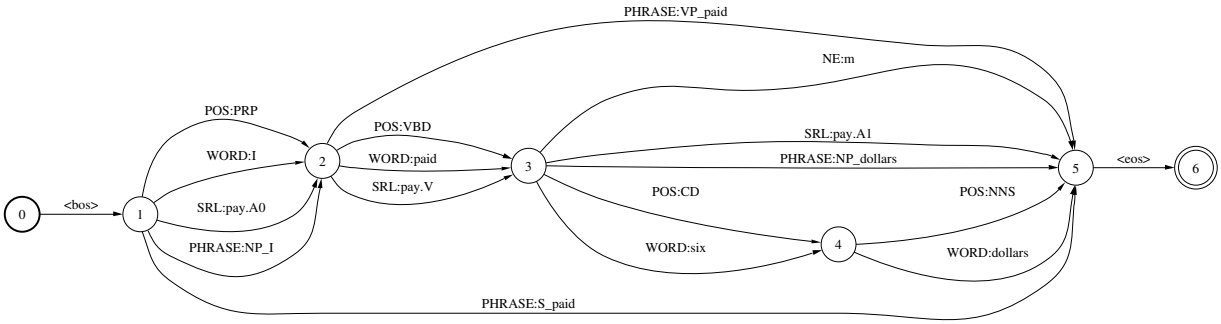

Figure 2: The SSG for the utterance *I paid six dollars*, where words (*WORD:*), part-of-speech tags (*POS:*), syntactic parse (*PHRASE:*), named entities (*NE:*) and semantic roles (*SRL:*) are included.

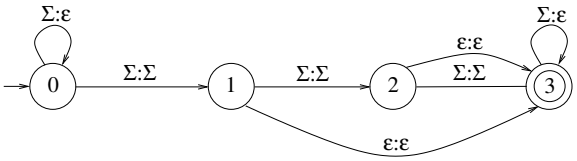

Figure 3: The FST used to extract unigram, bigram and trigrams. Σ represents the alphabet, ε represents the epsilon transition.

fore, *six dollars* is also represented by the transition labeled *PHRASE:NP_dollars*. As an alternative, one may drop the head word of the phrase from the representation, or insert an epsilon transition parallel to the transitions of the modifiers of the head word to eliminate them from some n-grams.

Generic named entity tags, such as person, location and organization names and task-dependent named entity tags, such as drug names in a medical domain, are also incorporated into the graph, where applicable. For instance, for the example sentence, *six dollars* is a monetary amount, so the arc *NE:m* is inserted parallel to that sequence.

As another source of semantic information, semantic role labels of the utterance components are incorporated to the SSGs. The semantic role labels represent the predicate/argument structure of each sentence: Given a predicate, the goal is to identify all of its arguments and their semantic roles. For example, in the example sentence the predicate is *pay*, the agent of this predicate is *I* and the amount is *six dollars*. In the graph, the labels of the transitions for semantic roles are prefixed by the token *SRL:* and the corresponding predicate. For example, the sequence *six dollars* is the amount of the predicate *pay*, and this is shown by the transition with label *SRL:pay.A1* following the PropBank notation (Kingsbury et al., 2002)[1].

In this work, we were only able to incorporate part-of-speech tags, syntactic parses, named entity tags and semantic role labels in the syntactic and semantic graphs. Insertion of further information such as supertags (Bangalore and Joshi, 1999) or word stems can also be beneficial for further processing.

3 Using SSGs for Call Classification

In this paper we propose extracting all n-grams from the SSGs to use them for call classification. The n-grams in an utterance SSG can be extracted by converting it to a finite state transducer (FST), F_i. Each transition of F_i has the labels of the arcs on the SSG as input and output symbols[2]. Composing this FST with another FST, F_N, representing all the possible n-grams, forms the FST, F_X, which includes all n-grams in the SSG:

$$F_X = F_i \circ F_N$$

[1] *A1* or *Arg1* indicates the object of the predicate, in this case the amount.

[2] Instead of the standard notation where ":" is used to separate the input and output symbols in finite state transducers, we use ":" to separate the type of the token and its value.

Then, extracting the n-grams in the SSG is equivalent to enumerating all paths of F_X. For $n = 3$, F_N is shown in Figure 3. The alphabet Σ contains all the symbols in F_i.

We expect the SSGs to help call classification because of the following reasons:

- First of all, the additional information is expected to provide some generalization, by allowing new n-grams to be encoded in the utterance graph since SSGs provide syntactic and semantic groupings. For example, the words *a* and *the* both have the part-of-speech tag category *DT* (determiner), or all the numbers are mapped to a cardinal number (*CD*), like the *six* in the example sentence. So the bigrams *WORD:six WORD:dollars* and *POS:CD WORD:dollars* will both be in the SSG. Similarly the sentences *I paid six dollars* and *I paid seventy five dollars and sixty five cents* will both have the trigram *WORD:I WORD:paid NE:m* in their SSGs.
- The head words of the syntactic phrases and predicate of the arguments are included in the SSGs. This enables the classifier to handle long distance dependencies better than using other simpler methods, such as extracting all gappy n-grams. For example, consider the following two utterances: *I need a copy of my bill* and *I need a copy of a past due bill*. As shown in Figures 4 and 5, the n-gram *WORD:copy WORD:of PHRASE:NP_bill* appears for both utterances, since both subsequences *my bill* and *a past due bill* are nothing but noun phrases with the head word *bill*.
- Another motivation is that, when using simply the word n-grams in an utterance, the classifier is only given lexical information. Now the classifier is provided with more and different information using these extra syntactic and semantic features. For example, a named entity of type monetary amount may be strongly associated with some call-type.
- Furthermore, there is a close relationship between the call-types and semantic roles. For example, if the predicate is *order* this is most probably the call-type *Order(Item)* in a retail domain application. The simple n-gram approach would consider all the appearances of the unigram *order* as equal. However consider the utterance *I'd like to check an order* of a different call-type, where the *order* is not a predicate but an object. Word n-gram features will fail to capture this distinction.

Once the SSG of an utterance is formed, all the n-grams are extracted as features, and the decision of which one to select/use is left to the classifier.

4 Computation of the SSGs

In this section, the tools used to compute the information in SSGs are described and their performances on manually transcribed spoken dialog utterances are presented. All of these components may be improved independently, for the specific application domain.

4.1 Part-of-Speech Tagger

Part-of-speech tagging has been very well studied in the literature for many languages, and the approaches vary from rule-based to HMM-based and classifier-based (Church, 1988; Brill, 1995, among others) tagging. In our framework, we employ a simple HMM-based tagger, where the most probable tag sequence, $\hat{T}$, given the words, W, is output (Weischedel et al., 1993):

$$\hat{T} = \underset{T}{\operatorname{argmax}}\, P(T|W) = \underset{T}{\operatorname{argmax}}\, P(W|T)P(T)$$

Since we do not have enough data which is manually tagged with part-of-speech tags for our applications, we used Penn Treebank (Marcus et al., 1994) as our training set. Penn Treebank includes data from Wall Street Journal, Brown, ATIS, and Switchboard corpora. The final two sets are the most useful for our domain, since they are also from spoken language and include disfluencies. As a test set, we manually labeled 2,000 words of user utterances from an AT&T VoiceTone spoken dialog system application, and we achieved an accuracy of 94.95% on manually transcribed utterances. When we examined the errors, we have seen that the frequent word *please* is mis-labeled or frequently occurs as a verb in the training data, even when it is not. Given that the latest literature on POS tagging using Penn Treebank reports an accuracy of around 97% with in-domain

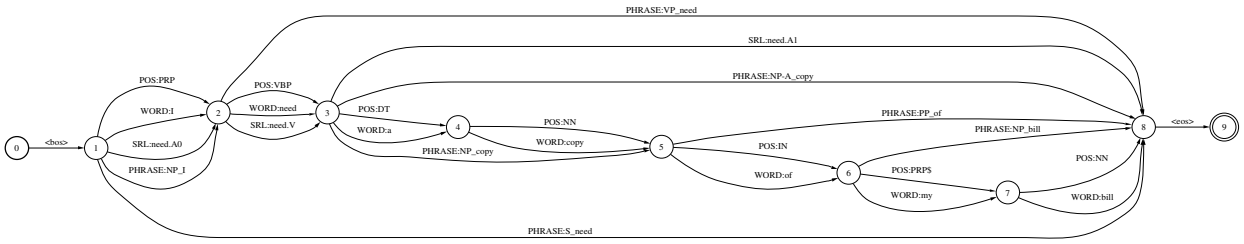

Figure 4: An example SSG for the utterance *I need a copy of my bill*.

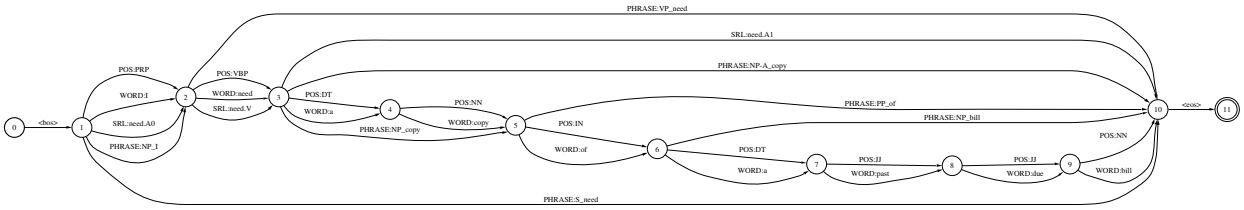

Figure 5: An example SSG for the utterance *I need a copy of a past due bill*.

training data (van Halteren et al., 2001), we achieve a very reasonable performance, considering these errors.

4.2 Syntactic Parser

For syntactic parsing, we use the Collins' parser (Collins, 1999), which is reported to give over 88% labeled recall and precision on Wall Street Journal portion of the Penn Treebank. We use Buchholz's `chunklink` script to extract information from the parse trees[3]. Since we do not have any data from our domain, we do not have a performance figure for this task for our domain.

4.3 Named Entity Extractor

For named entity extraction, we tried using a simple HMM-based approach, a simplified version of BBN's name finder (Bikel et al., 1999), and a classifier-based tagger using Boostexter (Schapire and Singer, 2000). In the simple HMM-based approach, which is the same as the part-of-speech tagging, the goal is to find the tag sequence, $\hat{T}$, which maximizes $P(T|W)$ for the word sequence, W. The tags in this case are named entity categories (such as *P* and *p* for Person names, *O* and *o* for Organization names, etc. where upper-case indicates the first word in the named entity) or *NA* if the word is not a part of a named entity. In the simplified version of BBN's name finder, the states of the model were word/tag combinations, where the tag t_i for word w_i is the named entity category of each word. Transition probabilities consisted of trigram probabilities $P(w_i/t_i|w_{i-1}/t_{i-1}, w_{i-2}/t_{i-2})$ over these combined tokens. In the final version, we extended this model with an unknown words model (Hakkani-Tür et al., 1999). In the classifier-based approach, we used simple features such as the current word and surrounding 4 words, binary tags indicating if the word considered contains any digits or is formed from digits, and features checking capitalization (Carreras et al., 2003).

To test these approaches, we have used data from an AT&T VoiceTone spoken dialog system application for a pharmaceutical domain, where some of the named entity categories were person, organization, drug name, prescription number, and date. The training and test sets contained around 11,000 and 5,000 utterances, respectively. Table 1 summarizes the overall F-measure results as well as F-measure for the most frequent named entity categories. Overall, the classifier based approach resulted in the best performance, so it is also used for the call classification experiments.

4.4 Semantic Role Labeling

The goal of semantic role labeling is to extract all the constituents which fill a semantic role of a target verb. Typical semantic arguments include Agent, Patient, Instrument, etc. and also adjuncts such as Locative, Temporal, Manner, Cause, etc. In this

[3] http://ilk.kub.nl/∼sabine/chunklink/chunklink_2-2-2000_for_conll.pl

Category	Count	HMM	IF	Boostexter
Org.	132	62.0	73.8	70.9
Person	150	45.0	62.4	54.4
Date	178	51.4	61.9	72.0
Drug	220	65.7	62.3	63.1
Overall	836	54.5	56.8	64.0

Table 1: F-Measure results for named entity extraction with various approaches. HMM is the simple HMM-based approach, IF is the simplified version of BBN's name finder with an unknown words model.

work, we use the semantic roles and annotations from the PropBank corpus (Kingsbury et al., 2002), where the arguments are given mnemonic names, such as Arg0, Arg1, Arg-LOC, etc. For example, for the sentence *I have bought myself a blue jacket from your summer catalog for twenty five dollars last week*, the agent (buyer, or Arg0) is *I*, the predicate is *buy*, the thing bought (Arg1) is *a blue jacket*, the seller or source (Arg2) is *from your summer catalog*, the price paid (Arg3) is *twenty five dollars*, the benefactive (Arg4) is *myself*, and the date (ArgM-TMP) is *last week*[4].

Semantic role labeling can be viewed as a multi-class classification problem. Given a word (or phrase) and its features, the goal is to output the most probable semantic label. For semantic role labeling, we have used the exact same feature set that Hacioglu *et al.* (2004) have used, since their system performed the best among others in the CoNLL-2004 shared task (Carreras and Màrquez, 2004). We have used Boostexter (Schapire and Singer, 2000) as the classifier. The features include token-level features (such as the current (head) word, its part-of-speech tag, base phrase type and position, etc.), predicate-level features (such as the predicate's lemma, frequency, part-of-speech tag, etc.) and argument-level features which capture the relationship between the token (head word/phrase) and the predicate (such as the syntactic path between the token and the predicate, their distance, token position relative to the predicate, etc.).

In order to evaluate the performance of semantic role labeling, we have manually annotated 285 utterances from an AT&T VoiceTone spoken dialog system application for a retail domain. The utterances include 645 predicates (2.3 predicates/utterance). First we have computed recall and precision rates for evaluating the predicate identification performance. The precision is found to be 93.04% and recall is 91.16%. More than 90% of false alarms for predicate extraction are due to the word *please*, which is very frequent in customer care domain and erroneously tagged as explained above. Most of the false rejections are due to disfluencies and ungrammatical utterances. For example in the utterance *I'd like to order place an order*, the predicate *place* is tagged as a noun erroneously, probably because of the preceding verb *order*. Then we have evaluated the argument labeling performance. We have used a stricter measure than the CoNLL-2004 shared task. The labeling is correct if both the boundary and the role of all the arguments of a predicate are correct. In our test set, we have found out that our SRL tool correctly tags all arguments of 57.4% of the predicates.

5 Call Classification Experiments and Results

In order to evaluate our approach, we carried out call classification experiments using human-machine dialogs collected by the spoken dialog system used for customer care. We have only considered utterances which are responses to the greeting prompt *How may I help you?* in order not to deal with confirmation and clarification utterances. We first describe this data, and then give the results obtained by the semantic classifier. We have performed our tests using the Boostexter tool, an implementation of the Boosting algorithm, which iteratively selects the most discriminative features for a given task (Schapire and Singer, 2000).

5.1 Data

Table 2 summarizes the characteristics of our application including the amount of training and test data, total number of call-types, average utterance length, and call-type perplexity. Perplexity is computed using the prior distribution over all the call-types in the training data.

5.2 Results

For call classification, we have generated SSGs for the training and test set utterances using the tools

[4]See http://www.cis.upenn.edu/∼dgildea/Verbs for more details

Training Data Size	3,725 utterances
Test Data Size	1,954 utterances
Number of Call-Types	79
Call-Type Perplexity	28.86
Average Utterance Length	12.08 words

Table 2: Characteristics of the data used in the experiments.

	Baseline	Using SSG	Increase
Unigram	2,303	6,875	2.99 times
Bigram	15,621	112,653	7.21 times
Trigram	34,185	705,673	20.64 times
Total	52,109	825,201	15.84 times

Table 3: A comparison of number of features.

described above. When n-grams are extracted from these SSGs, instead of the word graphs (Baseline), there is a huge increase in the number of features given to the classifier, as seen in Table 3. The classifier has now 15 times more features to work with. Although one can apply a feature selection approach before classification as frequently done in the machine learning community, we left the burden of analyzing 825,201 features to the classifier.

Table 4 presents the percentage of the features selected by Boostexter using SSGs for each information category. As expected the lexical information is the most frequently used, and 54.06% of the selected features have at least one word in its n-gram. The total is more than 100%, since some features contain more than one category, as in the bigram feature example: *POS:DT WORD:bill*. This shows the use of other information sources as well as words.

Table 5 presents our results for call classification. As the evaluation metric, we use the top class error rate (TCER), which is the ratio of utterances, where the top scoring call-type is not one of the true call-types assigned to each utterance by the human labelers. The baseline TCER on the test set using only word n-grams is 23.80%. When we extract features from the SSGs, we see a 2.14% relative decrease in the error rate down to 23.29%. When we analyze these results, we have seen that:

- For "easy to classify" utterances, the classifier already assigns a high score to the true call-type using just word n-grams.
- The syntactic and semantic features extracted from the SSGs are not 100% accurate, as presented earlier. So, although many of these features have been useful, there is certain amount of noise introduced in the call classification training data.
- The particular classifier we use, namely Boosting, is known to handle large feature spaces poorer than some others, such as SVMs. This is especially important with 15 times more features.

Category		Frequency
Lexical	Words	54.06%
Syntactic	Part-of-Speech	49.98%
	Syntactic Parse	27.10%
Semantic	Named Entity	1.70%
	Semantic Role Label	11.74%

Table 4: The percentage of the features selected by the classifier for each information category

	Baseline	SSGs	Decrease
All utterances	23.80%	23.29%	2.14%
Low confidence utterances	68.77%	62.16%	9.61%
All utterances (Cascaded)	23.80%	22.67%	4.74%

Table 5: Call classification error rates using words and SSGs.

Due to this analysis, we have focused on a subset of utterances, namely utterances with low confidence scores, i.e. cases where the score given to the top scoring call-type by the baseline model is below a certain threshold. In this subset we had 333 utterances, which is about 17% of the test set. As expected the error rates are much higher than the overall and we get much larger improvement in performance when we use SSGs. The baseline for this set is 68.77%, and using extra features, this reduces to 62.16% which is a 9.61% relative reduction in the error rate.

This final experiment suggests a cascaded approach for exploiting SSGs for call classification.

That is, first the baseline word n-gram based classifier is used to classify all the utterances, then if this model fails to commit on a call-type, we perform extra feature extraction using SSGs, and use the classification model trained with SSGs. This cascaded approach reduced the overall error rate of all utterances from 23.80% to 22.67%, which is 4.74% relative reduction in error rate.

6 Conclusions

In this paper, we have introduced syntactic and semantic graphs (SSGs) for speech and language processing. We have described their use for the task of call classification. We have presented results showing 4.74% improvement, using utterances collected from AT&T VoiceTone spoken dialog system. SSGs can also be useful for text classification and other similar language processing applications. Our future work includes feature selection prior to classification and developing methods that are more robust to ASR errors while computing the SSGs. We also plan to improve the syntactic and semantic processing components by adapting the models with some amount of labeled in-domain spoken dialog data.

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2), June.

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning Journal Special Issue on Natural Language Learning*, 34(1-3):211–231.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565, December.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, Boston, MA, May.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. 2003. A simple named entity extractor using AdaBoost. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, Edmonton, Canada.

Jennifer Chu-Carroll and Bob Carpenter. 1999. Vector-based natural language call routing. *Computational Linguistics*, 25(3):361–388.

Kenneth W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing (ANLP)*, pages 136–143, Austin, Texas.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Computer and Information Science, Philadelphia, PA.

Giuseppe Di Fabbrizio , Dawn Dutton, Narendra Gupta, Barbara Hollister, Mazin Rahim, Giuseppe Riccardi, Robert Schapire, and Juergen Schroeter. 2002. AT&T help desk. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Denver, CO, September.

Allen L. Gorin, Giuseppe Riccardi, and Jerry H. Wright. 1997. How May I Help You? . *Speech Communication*, 23:113–127.

Narendra Gupta, Gokhan Tur, Dilek Hakkani-Tür, Srinivas Bangalore, Giuseppe Riccardi, and Mazin Rahim. To appear. The AT&T spoken language understanding system. *IEEE Transactions on Speech and Audio Processing*.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H. Martin, and Dan Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, Boston, MA, May.

Dilek Hakkani-Tür, Gokhan Tur, Andreas Stolcke, and Elizabeth Shriberg. 1999. Combining words and prosody for information extraction from speech. In *Proceedings of the EUROSPEECH'99,* 6^{th} *European Conference on Speech Communication and Technology*, Budapest, Hungary, September.

Paul Kingsbury, Mitch Marcus, and Martha Palmer. 2002. Adding semantic annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference (HLT)*, San Diego, CA, March.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Giuseppe Riccardi and Dilek Hakkani-Tür. 2003. Active and unsupervised learning for automatic speech recognition. In *Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH)*, Geneva, Switzerland, September.

Robert E. Schapire and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 2001. Improving accuracy in word class tagging through combination of machine learning systems. *Computational Linguistics*, 27(2):199–230.

Ralph Weischedel, Richard Schwartz, Jeff Palmucci, Marie Meteer, and Lance Ramshaw. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics, Special Issue on Using Large Corpora*, 19(2):361–382, June.