

# Statistical Shallow Semantic Parsing despite Little Training Data

**Rahul Bhagat**

Information Sciences  
Institute  
University of Southern  
California  
Marina del Rey,  
CA, 90292, USA  
rahul@isi.edu

**Anton Leuski**

Institute for Creative  
Technologies  
University of Southern  
California  
Marina del Rey,  
CA, 90292, USA  
leuski@ict.usc.edu

**Eduard Hovy**

Information Sciences  
Institute  
University of Southern  
California  
Marina del Rey,  
CA, 90292, USA  
hovy@isi.edu

## Abstract

Natural Language Understanding is an essential module in any dialogue system. To achieve satisfactory performance levels, real time dialogue systems need the understanding module to be fast, accurate, and robust. Especially given the difficulty of obtaining sufficient training data, building a system that satisfies all these requirements is a hard problem. In this paper, we propose and compare a number of statistical learning based approaches to language understanding for a dialogue system. All the approaches result in systems that perform reasonably well despite being trained with very little data.

## 1 Introduction and Related Work

With the availability of semantic resources like FrameNet (Baker et al., 1998) and more recently PropBank (Kingsbury et al., 2002) a number of statistically trained semantic parsers have been reported in the recent past, including Gildea and Jurafsky (2002), Pradhan et al. (2004), Fleischman and Hovy (2003). All these parsers use statistical training approaches with interesting combinations of features to obtain reasonable performances on general purpose semantic-parsing tasks. However, despite their good general purpose capabilities, these parsers do not cover specific domains well, especially when these domains involve somewhat unusual terminology and rather detailed semantic out-

put. Where the general purpose parsers tend to provide case-frame structures, that include the standard core case roles (Agent, Patient, Instrument, etc.), dialogue oriented domains tend to require additional information about addressees, modality, speech acts, etc. Where general-purpose resources such as PropBank and Framenet provide invaluable training data for general case, it tends to be a problem to obtain enough training data in a specific dialogue oriented domain. We in this paper propose and compare a number of approaches for building a statistically trained domain specific parser/natural language understanding system (NLU) for a dialogue system.

Our NLU system is a part of Mission Rehearsal Exercise (MRE) project (Swartout et al., 2001), a large system that is being built to train experts, in which a trainee interacts with a Virtual Human using voice input. The virtual human includes speech recognition, language generation, and speech synthesis modules as well as a dialogue reasoner that interacts with other components that model the task status, trainee characteristics, and so on. The purpose of our NLU parser is to convert the sentence strings produced by the speech recognizer into internal shallow semantic frames for the dialogue module. Our engine has to operate in real time, has to be easily ported to new domains, and has to be robust against the somewhat unpredictable and noisy output of the speech recognizer.

These requirements are by no means trivial. The noisy output of the speech recognizer and the ability to operate in real time means that our systems cannot use any of the available statistically trained syntactic parsers (Charniak, 1997; Collins, 1997) to

obtain syntactic features for an input sentence. The reasons for this are the two-fold problems with the state of art syntactic parsers: (1) their inability to operate in real time; (2) their inability to parse the noisy speech output correctly. This leaves us with only a sparse feature set of words/ngrams for a sentence to be used to train our parsers. This makes the problem much harder.

The rest of the paper is organized as follows: Section 2 describes the input representation, input data and classes, Section 3 gives a detailed description of the different parsing methods, Section 4 describes the experimental set-up, Section 5 presents the individual results as well as a comparison of the different methods, and Section 6 discusses some future directions that we wish to pursue.

## 2 Representation, Data and Classes

In this section we discuss the format of the data that we use for the parser and the classes of entities used in our domain.

### 2.1 Representation

The parser output representation is a (possible nested) frame that consists of a set of slot-value pairs; each pair we call a ‘frame element’ or ‘meaning element’. For convenience, we collapse the nested frame representation into a flat representation in which the attributes are composed in sequence, as shown:

<b>Sentence:</b> clear the landing zone sergeant	
<b>Corresponding Frame:</b>	
<i>Nested form:</i>	<i>Collapsed form:</i>
s [	s.addresssee sgt
:addressee sgt	s.mood imperative
:mood imperative	
:sem [	s.sem.type action
:type action	s.sem.event clear
:event clear	s.sem.patient LZ
:patient LZ	s.sem.time present
:time present	
]]	

The values are represented in a lexicon and their superclasses in an ontology (see next section).

### 2.2 Data

Our training data is a parallel corpus of 477 sentences with manually created semantic frames. Our test set contains 50 sentence-frame pairs, some of which include words that are not present in the lexicon and/or have not been seen in the training data but are present in the lexicon.

### 2.3 Classes

Our very limited training data poses a problem. To overcome this problem to some extent, we group lexical items into an ontology of classes; any entities that fall into any of these classes are substituted by the corresponding class name in the training data. The classes are:

<b>AddresseeName</b> (generally a person) : johnson   tucci   sergeant   corporal   combat lifesaver . . .
<b>Location</b> : assembly area   area   Celic   LZ . . .
<b>Squad</b> (organization) : first squad   second squad   third squad   squad one . . .

## 3 Parsing Methods

### 3.1 Voting Model

We use a simple conditional probability model  $P(f | W)$  for parsing. The model represents the probability of producing slot-value pair  $f$  as an output given that we have seen a particular word or n-gram  $W$  as input. We use the collapsed frame form, thereby recording the nesting of meaning elements. We combine three such models in the system: unigram-based, bigram-based, and trigram-based. Let  $C(f_i | w_j)$  be the number of times meaning element  $f_i$  is seen as output given that we have seen unigram  $w_j$  anywhere in the input sentence. Similarly, let  $C(f_i | w_{j-1}w_j)$  be the number of times  $f_i$  is seen as output with bigram  $w_{j-1}w_j$  anywhere in the input and  $C(f_i | w_{j-2}w_{j-1}w_j)$  be the corresponding number with a trigram input.

a) Unigram Model:

$$P(f_i | w_j) = \frac{C(f_i | w_j)}{\sum_{k=1}^n C(f_k | w_j)}$$

b) Bigram Model:

$$P(f_i | w_{j-1}w_j) = \frac{C(f_i | w_{j-1}w_j)}{\sum_{k=1}^n C(f_k | w_{j-1}w_j)}$$

c) Trigram Model:

$$P(f_i | w_{j-2}w_{j-1}w_j) = \frac{C(f_i | w_{j-2}w_{j-1}w_j)}{\sum_{k=1}^n C(f_k | w_{j-2}w_{j-1}w_j)}$$

These models record the probability of correspondence of a word/ngram and a meaning element. Other than immediate ngram adjacency, the models record no long-distance dependencies between words and no dependencies among meaning elements. Despite this extreme simplicity, the models perform surprisingly well.

Our two-stage procedure for generating a frame for a given input sentence is: first find candidate meaning elements for each word/ngram input, and then select/filter from among all the candidates.

**Finding Candidates:** Given a string of words, this step finds a set of all the meaning elements that correspond with each word/ngram and hence may be a part of the final frame. We first replace all words/ngrams in the input sentence that belong to one of the three classes (Section 2.3) by their corresponding class labels and store them. We then generate a set of candidate meaning elements separately for each unigram, bigram, and trigram of the input sentence, each with an associated conditional probability, using the  $P(f | W)$  tables.

The set of candidates obtained after using either of the above models will of course contain many duplicate meaning elements, as different words/ngrams may ‘vote’ for the same candidates. As discussed in (Feng and Hovy, 2003), we simply add all the probability scores a candidate meaning element receives and assign this as the candidate’s weight:

$$\begin{aligned} Wt(f_i) &= \sum_{j=1}^n P(f_i | w_j) + \sum_{j=2}^n P(f_i | w_{j-1}w_j) \\ &+ \sum_{j=3}^n P(f_i | w_{j-2}w_{j-1}w_j) \end{aligned}$$

**Filtering:** In this step we select the most probable meaning elements from among the candidates to obtain the final frame for an input sentence.

We have found that a multi-level filtering scheme for selecting the final frame is key for improving accuracy and speed. This step-by-step procedure uses carefully designed heuristics to reduce the set of candidates down to the final frame.

**Level 1:** The first filter is a heuristic based on the observation that two meaning elements with the same attribute almost never occur in the same frame. (Here, recording frame nesting in the tables pays off.) For each attribute from the candidates, this filter chooses only the meaning element with highest weight.

**Level 2:** The second filter is also a heuristic, based on the observation that in this domain two meaning elements with the same value seldom occur in the same frame. This filter further reduces the list of candidates by choosing only the meaning element with the highest weight for each value.

**Level 3:** The third filter is a cutoff. It selects the top portion of the remaining candidates (based on a cutoff value) to form the complete frame.

### 3.2 Maximum Entropy

To explore other approaches to the problem than the simple voting based approach (Section 3.1), we cast our problem as a problem of ranking using a classifier.

In this setting, each slot-value pair occurring in the training data is considered a class. The feature set consists of the unigrams, bigrams and trigrams in the training data. The corresponding feature values are the product of the term frequencies (tf) and inverse document frequency (idf)— $tf*idf$  for each unigram, bigram and trigram in the training data. The classifier is trained on the entire training data. The task of building a frame for a sentence then reduces to selecting the top-n classes preferred by an input feature vector.

We first use the Maximum Entropy (Berger et al., 1996; Ratnaparkhi, 1996) classification approach. For conducting our experiments, we use the MEGA (Daumé III, 2004) Maximum Entropy classification package.

Our procedure for generating the frame again is a two step process as in Section 3.1, differing mainly in the way the set of candidates is generated.

**Finding Candidates:** Given a string of words, we obtain all unigrams, bigrams, and trigrams and their corresponding  $tf*idf$  feature values. We give this feature vector as input to MEGA, which as its output produces a ranked list of slot-value classes that this feature vector may correspond to along with their corresponding weights.

Filtering: The filtering step then is the same as in Section 3.1.

### 3.3 Support Vector Machines

In Section 3.2 we showed how we cast our problem as a classification task. So we decided to use another commonly used classifier, Support Vector Machine (Burges, 1998), to perform the same task.

For our experiments, we use the libsvm (Chang and Lin, 2005) package. We use libsvm in multiclass classification mode which uses the “one-against-one” approach (Chang and Lin, 2005) for multiclass classification. Also the libsvm option for “probability estimates” is set, so that it is trained to produce probability estimates for each class (which generates a ranked list of classes), instead of just the class label. The system uses a linear kernel with the cost parameter  $C$  set to 0.125. The classes and feature vectors are the same as for the Maximum Entropy classifier (Section 3.2).

With the above parameter settings, our two step procedure for generating a frame is then exactly the same as in Section 3.2.

### 3.4 Language Model

In Section 3.1 we showed how we estimate the probability of observing a particular slot-value pair given that we have observed a set of terms (i.e., unigrams, bigrams, and trigrams). We can approach this problem also from the perspective of language models. Let  $W = w_1, \dots, w_n$  be the observed utterance, let  $\mathcal{F}$  be the set of all slot-value pairs, and let  $P(f|W)$  be the probability of using a particular slot-value pair  $f \in \mathcal{F}$  to interpret the utterance  $W$ . In statistical language modeling (Ponte and Croft, 1997), word occurrences are considered to be random events and their probability is given by some distribution  $M$  which is called the *language model*. If we can estimate the language model for the slot-value pairs, then we can construct our target interpretation  $F$  as a set of the most likely slot-value pairs. Thus, our algorithm has two steps: (1) order all  $f \in \mathcal{F}$  by their  $P(f|W)$  and (2) select the top portion of the ordering to form the final frame  $F$ .

We construct the conditional probability  $P(f|W)$  using the joint probability of observing the slot-value pair  $f$  together with utterance words  $w_1, \dots, w_n$ :

$$P(f|W) = \frac{P(f, w_1, \dots, w_n)}{P(w_1, \dots, w_n)}$$

The problem of estimating the joint probability  $P(x_1, \dots, x_n)$  of several words occurring together to form a string of text  $X$  has received a lot of attention in recent years among researchers in the information retrieval community. The main challenge is to take into account the interdependencies that exist among the individual words while still making the computation feasible. Several different methods were suggested starting from the most trivial technique where all words assumed to be distributed identically and independently from each other:  $P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i)$ . Other approaches include Probabilistic Latent Semantic Indexing (PLSI) (Hoffman, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003), where the authors model text collections by a finite set of  $k$  topics and the joint probability is viewed as a mixture of the individual topic language models.

Lavrenko (Lavrenko, 2004) offers a more general approach where the word interdependencies are defined by an unknown parameter vector  $\theta$  and the words are taken as conditionally independent—they are independent for a given instance of the  $\theta$  vector. With the help of the de Finetti’s theorem, he showed that in this case the joint distribution can be represented as follows:

$$P(x_1, \dots, x_n) = \int_{\theta \in \Theta} \left\{ \prod_{i=1}^n P_{\theta}(x_i) \right\} p(\theta)$$

The variable  $\theta$  is the vector of hidden parameters,  $\Theta$  is the set of all possible parameter settings. Each  $P_{\theta}(x_i)$  is the appropriate probability distribution for individual words. The quantity  $p(\theta)$  is a probability measure that tells us which parameter vector  $\theta$  is a priori more likely. The author gives several approximations for that expression for different  $\Theta$ ,  $P(x)$ , and  $p(\cdot)$ . One of them is of particular interest to us.

Given a set of training strings  $S$ , the joint distribution can be approximated as

$$P(x_1, \dots, x_n) = \frac{1}{|S|} \sum_{s \in S} \prod_{i=1}^m \pi_s(x_i) \quad (1)$$

where  $|S|$  is the size of the training set and  $\pi_s(x_i)$  is the empirical probability distribution of words in

string  $s$ . It is estimated by the relative term frequency with an additional smoothing factor:

$$\pi_s(x) = \lambda_\pi \frac{\#(x, s)}{|s|} + (1 - \lambda_\pi) \frac{\sum_s \#(x, s)}{\sum_s |s|}$$

where  $\#(x, s)$  is the number of times word  $x$  appears in string  $s$ ,  $|s|$  is the length of the string  $s$ , and the constant  $\lambda_\pi$  is the tunable parameter that can be determined from the training data.

Equation 1 assumes that all words  $x_i$  came from the same vocabulary. We can show that in the case of two different vocabularies, the joint distribution has the following form:

$$P(f, w_1, \dots, w_n) = \frac{1}{|S|} \sum_{s \in \{F_s, W_s\}} \phi_{F_s}(f) \prod_{i=1}^m \pi_{W_s}(w_i) \quad (2)$$

Here  $s$  iterates over the set of training pairs that maps an utterance  $W_s$  to its frame interpretation  $F_s$ .  $\phi_F(f)$  is the empirical probability distribution of slot-value pairs in frame  $F$ :

$$\phi_F(f) = \lambda_\phi \frac{\#(f, F)}{|F|} + (1 - \lambda_\phi) \frac{\sum \#(f, F)}{\sum |F|}$$

Combining these estimations we get the following expression for  $P(f|W)$ :

$$P(f|W) = \frac{\sum_s \phi_{F_s}(f) \prod_{i=1}^m \pi_{W_s}(w_i)}{\sum_s \prod_{i=1}^m \pi_{W_s}(w_i)}$$

Note one problem with this approach: the words in the utterance are assumed to be exchangeable, e.g., sentences “the area is secured” and “is the area secured” will have the same probabilities, which may potentially lead to questions interpreted as statements and vice versa. We deal with the problem by including local order dependencies in  $\pi_s(w)$  in the form of a trigram model:

$$\begin{aligned} \pi_{3,s}(w) &= \lambda_1 \frac{\#(w_{-2}w_{-1}w, s)}{\#(w_{-2}w_{-1}, s)} \\ &+ \lambda_2 \frac{\sum_s \#(w_{-2}w_{-1}w, s)}{\sum_s \#(w_{-2}w_{-1}, s)} \\ &+ \lambda_3 \frac{\#(w_{-1}w, s)}{\#(w_{-1}, s)} + \lambda_4 \frac{\sum_s \#(w_{-1}w, s)}{\sum_s \#(w_{-1}, s)} \\ &+ \lambda_5 \frac{\#(w, s)}{|s|} + \lambda_6 \frac{\sum_s \#(w, s)}{\sum_s |s|} \end{aligned}$$

where  $\sum_i \lambda_i = 1$ . Here  $w_{-1}$  and  $w_{-2}$  are the words immediately preceding  $w$  in  $s$ .

## 4 Experimental Setup

As mentioned before, we train all our systems on a training set of 477 sentence-frame pairs. All the parameters for the different methods are optimized using cross validation on the training data. The systems are then tested on an unseen test set of 50 sentences. The sentences in the unseen test set are from the same domain, but do contain words not present in the training set and in some cases not even present in the lexicon.

For the test sentences, the frames generated by each system are compared against the manually built gold standard frames, and Precision, Recall and F-scores are calculated for each frame as follows:

$$Precision = \frac{\# \text{ of correct slot-value pairs by system}}{\# \text{ of slot-value pairs by system}}$$

$$Recall = \frac{\# \text{ of correct slot-value pairs by system}}{\# \text{ of slot-value pairs in gold-standard}}$$

$$F\text{-score} = \frac{2 * Precision * Recall}{Precision + Recall}$$

To compare the systems against each other, for each system we calculate an average Precision, Recall, and F-score. Also to test the statistical significance of these scores, we conduct a one-tailed Student’s t test (Manning and Schtze, 1999) on the F-scores of these systems for the 50 test cases. The results are reported below.

Finally, since we are training on very limited data, it is important to see the effect that the size of training data has on the performances of the different systems. To access this, we train our systems on  $\frac{1}{4}$  of the training data, then on  $\frac{1}{2}$  of the training data, then on  $\frac{3}{4}$ , and finally on the full training data. We record the average F-scores of the systems on the test data in each case and then plot the learning curves for the different systems.

## 5 Results

Table 1 shows the average Precision, Recall and F-scores of the different systems for the 50 test sentences: Voting based (Voting), Maximum Entropy based (ME), Support Vector Machine based (SVM), Language Model based with unigrams (LM1) and Language Model based with trigrams (LM2). The F-scores show that the LM2 system performs the best

Method	Precision	Recall	F-score
Voting	0.82	0.78	0.80
ME	0.77	0.80	0.78
SVM	0.79	0.72	0.75
LM1	0.80	0.84	0.82
LM2	0.82	0.84	0.83

Table 1: Performance of different systems on test data.

Training size	Voting	ME	SVM	LM1
$\frac{1}{4}$	0.6871	0.6483	0.6224	0.6936
$\frac{1}{2}$	0.7335	0.6885	0.6717	0.7403
$\frac{3}{4}$	0.7417	0.7215	0.7249	0.7688
Full	0.7855	0.7723	0.7399	0.8151

Table 2: Average F-scores of the systems on different sizes of training data.

though the system scores in general for all the systems are very close.

To test the statistical significance of these scores, we conduct a two-tailed paired Student's t test (Manning and Schtze, 1999) on the F-scores of these systems for the 50 test cases. The test shows that there is no statistically significant difference in their performances.

Finally, Table 3 shows the average F-scores of the different systems on the different amounts of training data<sup>1</sup>. Figure 1 shows the corresponding learning curves. The F-scores and learning curves clearly show that even small chunks of training data make

<sup>1</sup>F-score table for LM2 would be provided with the final submission.

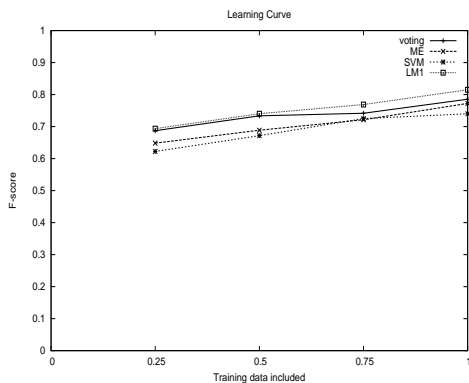


Figure 1: Learning curves for the different systems.

a noticeable though not a large difference in system performances. It is good news that we do not have to amass a large amount of training data before we can begin using the NLU.

## 6 Conclusions and Future Work

This work illustrates that one can achieve fair success in building a statistical NLU engine for a restricted domain using relatively little training data and surprisingly using a rather simple voting model. The consistently good results obtained from all the systems on the task clearly indicate the feasibility of using only word/ngram level features for parsing, and shows that the good performance obtained is not accidental. Using just word/ngram features not only speeds up the systems but also enables the parser to deal with ill-formed English sentences.

This work also points to the advantages that the statistical paradigm gives us over the rule based paradigm. First, it gives us the ability to experiment with different learning methods. Second, it makes the NLU very robust. Agreed, that none of the systems always produce 100% correct output, but none simply fail either when malformed input is provided. In our domain, many instances of malformed NLU outputs are not catastrophic, since even partial frames may fulfill some useful function in certain circumstances. Finally, it makes the parser easily portable. The merging of statistical learning methods and a lexicon/ontology has allowed us to port our NLU to a new domain in a rather short time. In under three months, the NLU was operational in a completely new domain.

The tiny amount of training data has been augmented by the inclusion of a lexicon and associated ontology of class labels. We are conducting experiments with out-of-lexicon words to determine whether the NLU can be extended to handle them if not completely then at least gracefully.

Also, we are planning to build a hybrid system that combines the outputs of the different approaches/methods in a way so as to take best of each approach.

Analysis of the problem and our experience while building the NLU has taught us several lessons. The comparable results obtained by using the different methods clearly indicate that getting any further per-

formance boost would either require a significant increase in the amount of training data or a detailed analysis of the problem cases. Simply using new learning algorithms is unlikely to help our cause. Since building more training data is a slow and painful process, we have identified a number of problem cases that we wish to deal with first. One major problem that has consistently hurt system performance is complex, multi-clause utterances. To take care of this problem, we are making efforts along with our colleagues in the speech community to build a real-time speech utterance-chunker. This utterance-chunker would try to use acoustic as well as linguistic cues to break a multi-clause utterance into smaller phrase-length chunks that the NLU can deal with gracefully. We are eager to discover any performance benefits.

Finally, this work leads us to believe that even though the problem of building a good Natural Language Understanding system is far from solved yet, one can obtain reasonable results, despite relatively little training data, with a variety of statistically trained approaches.

## References

- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of COLING/ACL*, page 8690, Montreal, Canada.
- Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.
- D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.
- Christopher J. C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.
- Chih-Chung Chang and Chih-Jen Lin, 2005. *LIBSVM: a library for support vector machines*.
- Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *AAAI/IAAI*, pages 598–603.
- Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 16–23, Morristown, NJ, USA. Association for Computational Linguistics.
- Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Notes available at <http://www.isi.edu/~hdaume>, August.
- Donghui Feng and Eduard Hovy. 2003. Semantics-oriented language understanding with automatic adaptability. In *Proceedings of Natural Language Processing and Knowledge Engineering*.
- Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.
- T. Hoffman. 1999. Probabilistic latent semantic indexing. In *Proceedings on the 22nd annual international ACM SIGIR conference*, pages 50–57.
- Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the penn treebank. In *Proceedings of HLT Conference*.
- Victor Lavrenko. 2004. *A Generative Theory of Relevance*. Ph.D. thesis, University of Massachusetts at Amherst.
- Christopher D. Manning and Hinrich Schtze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.
- Jay M. Ponte and W. Bruce Croft. 1997. Text segmentation by topic. In *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, pages 120–129.
- S. Pradhan, K. Hacioglu, W. Ward, J. Martin, and D. Jurafsky. 2003. Semantic role parsing: Adding semantic structure to unstructured text.
- Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL)*, Boston, MA.
- Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey.
- W. Swartout, R. Hill, J. Gratch, W. Johnson, C. Kyriakakis, C. LaBore, R. Lindheim, S. Marsella, D. Miraglia, B. Moore, J. Morie, J. Rickel, M. Thiebaux, L. Tuch, R. Whitney, and J. Douglas. 2001. Toward the holodeck: Integrating graphics, sound, character and story. In *Proceedings of Autonomous Agents*.