A Novel Use of Statistical Parsing to Extract Information from Text

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel
BBN Technologies
70 Fawcett Street, Cambridge, MA 02138
szmiller@bbn.com

Abstract

Since 1995, a few statistical parsing algorithms have demonstrated a breakthrough in parsing accuracy, as measured against the UPenn TREEBANK as a gold standard. In this paper we report adapting a lexicalized, probabilistic context-free parser to information extraction and evaluate this new technique on MUC-7 template elements and template relations.

1   Introduction

Since 1995, a few statistical parsing algorithms (Magerman, 1995; Collins, 1996 and 1997; Charniak, 1997; Rathnaparki, 1997) demonstrated a breakthrough in parsing accuracy, as measured against the University of Pennsylvania TREEBANK as a gold standard. Yet, relatively few have embedded one of these algorithms in a task. Chelba, (1999) was able to use such a parsing algorithm to reduce perplexity with the long term goal of improved speech recognition.

In this paper, we report adapting a lexicalized, probabilistic context-free parser with head rules (LPCFG-HR) to information extraction. The technique was benchmarked in the Seventh Message Understanding Conference (MUC-7) in 1998.

Several technical challenges confronted us and were solved:

How could the limited semantic interpretation required in information extraction be integrated into the statistical learning algorithm? We were able to integrate both syntactic and semantic information into the parsing process, thus avoiding potential errors of syntax first followed by semantics.

Would TREEBANKing of the variety of news sources in MUC-7 be required? Or could the University of Pennsylvania's TREEBANK on Wall Street Journal adequately train the algorithm for New York Times newswire, which includes dozens of newspapers? Manually creating source-specific training data for syntax was not required. Instead, our parsing algorithm, trained on the UPenn TREEBANK, was run on the New York Times source to create unsupervised syntactic training which was constrained to be consistent with semantic annotation.

Would semantic annotation require computational linguists? We were able to specify relatively simple guidelines that students with no training in computational linguistics could annotate.

2   Information Extraction Tasks

We evaluated the new approach to information extraction on two of the tasks of the Seventh Message Understanding Conference (MUC-7) and reported in (Marsh, 1998). The Template Element (TE) task identifies organizations, persons, locations, and some artifacts (rocket and airplane-related artifacts). For each organization in an article, one must identify all of its names as used in the article, its type (corporation, government, or other), and any significant description of it. For each person, one must find all of the person's names within the document, his/her type (civilian or military), and any significant descriptions (e.g., titles). For each location, one must also give its type (city, province, county, body of water, etc.). For the following example, the template element in Figure 1 was to be generated:

"...according to the report by Edwin Dorn, under secretary of defense for personnel and readiness. …Dorn's conclusion that Washington…"

```
<ENTITY-9601020516-13> :=
   ENT_NAME: "Edwin Dorn"
    "Dorn"
   ENT_TYPE: PERSON
   ENT_DESCRIPTOR: "under secretary of
     defense for personnel and readiness"
   ENT_CATEGORY: PER_CIV
```

Figure 1: An example of the information to be extracted for TE.

The Template Relations (TR) task involves identifying instances of three relations in the text:

the products made by each company

the employees of each organization,

the (headquarters) location of each organization.

TR builds on TE in that TR reports binary relations between elements of TE. For the following example, the template relation in Figure 2 was to be generated: "Donald M. Goldstein, a historian at the University of Pittsburgh who helped write…"

```
<EMPLOYEE_OF-9601020516-5> :=
    PERSON: <ENTITY-9601020516-18>
    ORGANIZATION: <ENTITY-
     9601020516-9>
<ENTITY-9601020516-9> :=
   ENT_NAME: "University of Pittsburgh"
   ENT_TYPE: ORGANIZATION
   ENT_CATEGORY: ORG_CO
<ENTITY-9601020516-18> :=
   ENT_NAME: "Donald M. Goldstein"
   ENT_TYPE: PERSON
   ENT_DESCRIPTOR: "a historian at the
   University of Pittsburgh"
```
Figure 2: An example of information to be extracted for TR

3 Integrated Sentential Processing

Almost all approaches to information extraction – even at the sentence level – are based on the divide-and-conquer strategy of reducing a complex problem to a set of simpler ones. Currently, the prevailing architecture for dividing sentential processing is a four-stage pipeline consisting of:

1. part-of-speech tagging

2. name finding

3. syntactic analysis, often limited to noun and verb group chunking

4. semantic interpretation, usually based on pattern matching

Since we were interested in exploiting recent advances in parsing, replacing the syntactic analysis stage of the standard pipeline with a modern statistical parser was an obvious possibility. However, pipelined architectures suffer from a serious disadvantage: errors accumulate as they propagate through the pipeline. For example, an error made during part-of-speech-tagging may cause a future error in syntactic analysis, which may in turn cause a semantic interpretation failure. There is no opportunity for a later stage, such as parsing, to influence or correct an earlier stage such as part-of-speech tagging.

An integrated model can limit the propagation of errors by making all decisions jointly. For this reason, we focused on designing an integrated model in which tagging, name-finding, parsing, and semantic interpretation decisions all have the opportunity to mutually influence each other.

A second consideration influenced our decision toward an integrated model. We were already using a generative statistical model for part-of-speech tagging (Weischedel *et al.* 1993), and more recently, had begun using a generative statistical model for name finding (Bikel *et al.* 1997). Finally, our newly constructed parser, like that of (Collins 1997), was based on a generative statistical model. Thus, each component of what would be the first three stages of our pipeline was based on

the same general class of statistical model. Although each model differed in its detailed probability structure, we believed that the essential elements of all three models could be generalized in a single probability model.

If the single generalized model could then be extended to semantic analysis, all necessary sentence level processing would be contained in that model. Because generative statistical models had already proven successful for each of the first three stages, we were optimistic that some of their properties – especially their ability to learn from large amounts of data, and their robustness when presented with unexpected inputs – would also benefit semantic analysis.

4 Representing Syntax and Semantics Jointly

Our integrated model represents syntax and semantics jointly using augmented parse trees. In these trees, the standard TREEBANK structures are augmented to convey semantic information, that is, entities and relations. An example of an augmented parse tree is shown in Figure 3. The five key facts in this example are:

"Nance" is the name of a person.

"A paid consultant to ABC News" describes a person.

"ABC News" is the name of an organization.

The person described as "a paid consultant to ABC News" is employed by ABC News.

The person named "Nance" and the person described as "a paid consultant to ABC News" are the same person.

Here, each "reportable" name or description is identified by a "-r" suffix attached to its semantic label. For example, "per-r" identifies "Nance" as a named person, and "per-desc-r" identifies "a paid consultant to ABC News" as a person description. Other labels indicate relations among entities. For example, the co-reference relation between "Nance" and "a paid

consultant to ABC News" is indicated by "per-desc-of." In this case, because the argument does not connect directly to the relation, the intervening nodes are labeled with semantics "-ptr" to indicate the connection. Further details are discussed in the section Tree Augmentation.

5 Creating the Training Data

To train our integrated model, we required a large corpus of augmented parse trees. Since it was known that the MUC-7 evaluation data would be drawn from a variety of newswire sources, and that the articles would focus on rocket launches, it was important that our training corpus be drawn from similar sources and that it cover similar events. Thus, we did not consider simply adding semantic labels to the existing Penn TREEBANK, which is drawn from a single source – the Wall Street Journal – and is impoverished in articles about rocket launches.

Instead, we applied an information retrieval system to select a large number of articles from the desired sources, yielding a corpus rich in the desired types of events. The retrieved articles would then be annotated with augmented tree structures to serve as a training corpus.

Initially, we tried to annotate the training corpus by hand marking, for each sentence, the entire augmented tree. It soon became painfully obvious that this task could not be performed in the available time. Our annotation staff found syntactic analysis particularly complex and slow going. By necessity, we adopted the strategy of hand marking only the semantics.

Figure 4 shows an example of the semantic annotation, which was the only type of manual annotation we performed.

To produce a corpus of augmented parse trees, we used the following multi-step training procedure which exploited the Penn TREEBANK
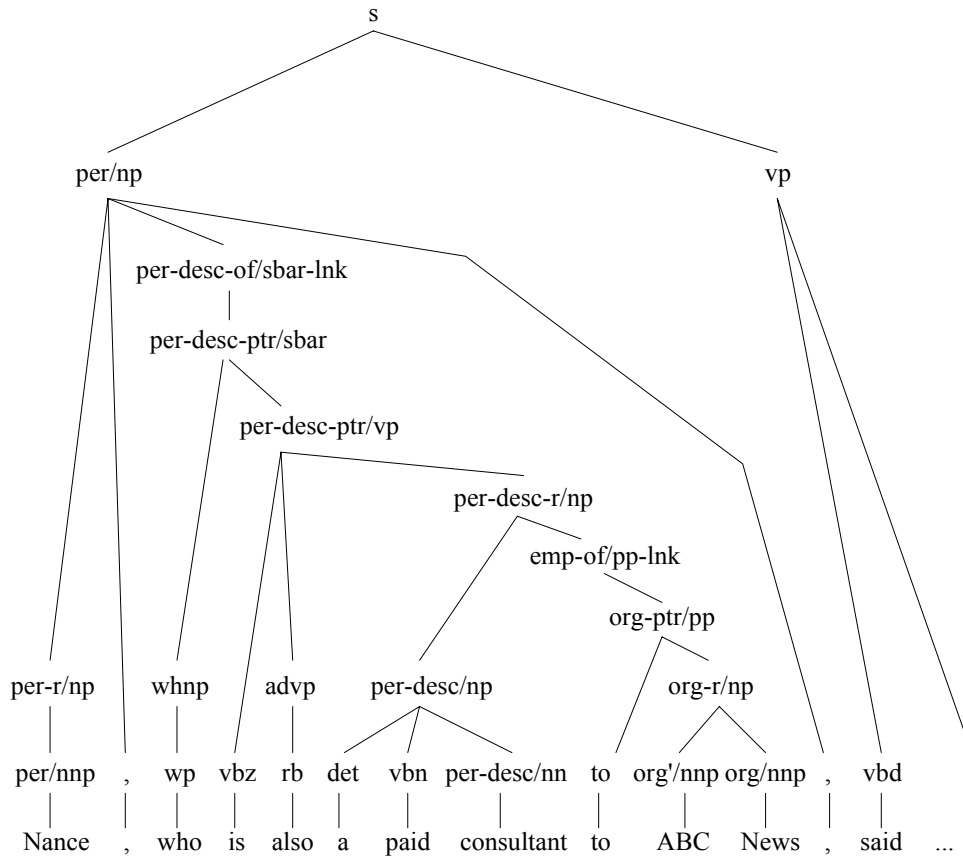
```
                                    s
                    ┌───────────────┴──────────────┐
                 per/np                            vp
       ┌──────────┬──────┴────────────────┐
       │   per-desc-of/sbar-lnk           │
       │          │                        │
       │   per-desc-ptr/sbar               │
       │          │                        │
       │     per-desc-ptr/vp               │
       │       ┌──┴──────────┐             │
       │       │        per-desc-r/np      │
       │       │              │            │
       │       │         emp-of/pp-lnk     │
       │       │              │            │
       │       │          org-ptr/pp       │
       │       │         ┌────┴────┐       │
     per-r/np  whnp advp per-desc/np  org-r/np
       │       │    │  ┌──┴──┬──────┐   ┌──┴──┐
     per/nnp , wp vbz rb det vbn per-desc/nn to org'/nnp org/nnp , vbd
       │    │  │   │   │  │   │       │      │    │        │     │   │
     Nance  , who is also a  paid consultant to  ABC     News   , said ...
```

Figure 3: An example of an augmented parse tree.

1. The model (see Section 7) was first trained on purely syntactic parse trees from the TREEBANK, producing a model capable of broad-coverage syntactic parsing.

2. Next, for each sentence in the semantically annotated corpus:

   a. The model was applied to parse the sentence, constrained to produce only parses that were consistent with the semantic annotation. A parse was considered consistent if no syntactic constituents crossed an annotated entity or description boundary.

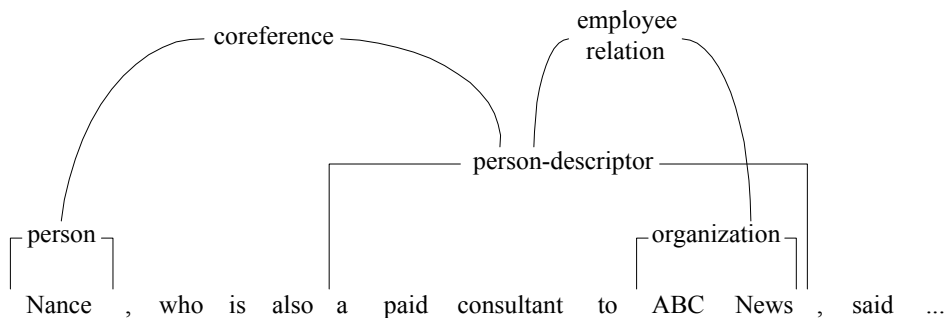   b. The resulting parse tree was then augmented to reflect semantic structure in addition to syntactic structure.

Figure 4: An example of semantic annotation.

Applying this procedure yielded a new version of the semantically annotated corpus, now annotated with complete augmented trees like that in Figure 3.

6 Tree Augmentation

In this section, we describe the algorithm that was used to automatically produce augmented trees, starting with a) human-generated semantic annotations and b) machine-generated syntactic parse trees. For each sentence, combining these two sources involved five steps. These steps are given below:

Tree Augmentation Algorithm

1. Nodes are inserted into the parse tree to distinguish names and descriptors that are not bracketed in the parse. For example, the parser produces a single noun phrase with no internal structure for "Lt. Cmdr. David Edwin Lewis". Additional nodes must be inserted to distinguish the description, "Lt. Cmdr.," and the name, "David Edwin Lewis."

2. Semantic labels are attached to all nodes that correspond to names or descriptors. These labels reflect the entity type, such as person, organization, or location, as well as whether the node is a proper name or a descriptor.

3. For relations between entities, where one entity is not a syntactic modifier of the other, the lowermost parse node that spans both entities is identified. A semantic tag is then added to that node denoting the relationship. For example, in the sentence "Mary Fackler Schiavo is the inspector general of the U.S. Department of Transportation," a co-reference semantic label is added to the *S* node spanning the name, "Mary Fackler Schiavo," and the descriptor, "the inspector general of the U.S. Department of Transportation."

4. Nodes are inserted into the parse tree to distinguish the arguments to each relation. In cases where there is a relation between two entities, and one of the entities is a

syntactic modifier of the other, the inserted node serves to indicate the relation as well as the argument. For example, in the phrase "Lt. Cmdr. David Edwin Lewis," a node is inserted to indicate that "Lt. Cmdr." is a descriptor for "David Edwin Lewis."

5. Whenever a relation involves an entity that is not a direct descendant of that relation in the parse tree, semantic *pointer labels* are attached to all of the intermediate nodes. These labels serve to form a continuous chain between the relation and its argument.

7 Model Structure

In our statistical model, trees are generated according to a process similar to that described in (Collins 1996, 1997). The detailed probability structure differs, however, in that it was designed to jointly perform part-of-speech tagging, name finding, syntactic parsing, and relation finding in a single process.

For each constituent, the head is generated first, followed by the modifiers, which are generated from the head outward. Head words, along with their part-of-speech tags and features, are generated for each modifier as soon as the modifier is created. Word features are introduced primarily to help with unknown words, as in (Weischedel *et al.* 1993).

We illustrate the generation process by walking through a few of the steps of the parse shown in Figure 3. At each step in the process, a choice is made from a statistical distribution, with the probability of each possible selection dependent on particular features of previously generated elements. We pick up the derivation just after the topmost *S* and its head word, *said*, have been produced. The next steps are to generate in order:

1. A head constituent for the S, in this case a *VP*.

2. Pre-modifier constituents for the *S*. In this case, there is only one: a *PER/NP*.

3. A head part-of-speech tag for the *PER/NP*, in this case *PER/NNP*.

4. A head word for the *PER/NP*, in this case *nance*.

5. Word features for the head word of the *PER/NP*, in this case *capitalized*.

6. A head constituent for the *PER/NP*, in this case a *PER-R/NP*.

7. Pre-modifier constituents for the *PER/NP*. In this case, there are none.

8. Post-modifier constituents for the *PER/NP*. First a comma, then an *SBAR* structure, and then a second comma are each generated in turn.

This generation process is continued until the entire tree has been produced.

We now briefly summarize the probability structure of the model. The categories for head constituents, c_h, are predicted based solely on the category of the parent node, c_p:

$$P(c_h \mid c_p), \text{ e.g. } P(vp \mid s)$$

Modifier constituent categories, c_m, are predicted based on their parent node, c_p, the head constituent of their parent node, c_{hp}, the previously generated modifier, c_{m-1}, and the head word of their parent, w_p. Separate probabilities are maintained for left (pre) and right (post) modifiers:

$$P_L(c_m \mid c_p, c_{hp}, c_{m-1}, w_p), \text{ e.g.}$$
$$P_L(per / np \mid s, vp, null, said)$$
$$P_R(c_m \mid c_p, c_{hp}, c_{m-1}, w_p), \text{ e.g.}$$
$$P_R(null \mid s, vp, null, said)$$

Part-of-speech tags, t_m, for modifiers are predicted based on the modifier, c_m, the part-of-speech tag of the head word, t_h, and the head word itself, w_h:

$$P(t_m \mid c_m, t_h, w_h), \text{ e.g.}$$
$$P(per / nnp \mid per / np, vbd, said)$$

Head words, w_m, for modifiers are predicted based on the modifier, c_m, the part-of-speech tag of the modifier word , t_m, the part-of-speech tag of the head word , t_h, and the head word itself, w_h:

$$P(w_m \mid c_m, t_m, t_h, w_h), \text{ e.g.}$$
$$P(nance \mid per / np, per / nnp, vbd, said)$$

Finally, word features, f_m, for modifiers are predicted based on the modifier, c_m, the part-of-speech tag of the modifier word , t_m, the part-of-speech tag of the head word , t_h, the head word itself, w_h, and whether or not the modifier head word, w_m, is known or unknown.

$$P(f_m \mid c_m, t_m, t_h, w_h, known(w_m)), \text{ e.g.}$$
$$P(cap \mid per / np, per / nnp, vbd, said, true)$$

The probability of a complete tree is the product of the probabilities of generating each element in the tree. If we generalize the tree components (constituent labels, words, tags, etc.) and treat them all as simply elements, e, and treat all the conditioning factors as the history, h, we can write:

$$P(tree) = \prod_{e \in tree} P(e \mid h)$$

8 Training the Model

Maximum likelihood estimates for the model probabilities can be obtained by observing frequencies in the training corpus. However, because these estimates are too sparse to be relied upon, we use interpolated estimates consisting of mixtures of successively lower-order estimates (as in Placeway *et al.* 1993).

For modifier constituents, the mixture components are:
$$\tilde{P}(c_m \mid c_p, c_{hp}, c_{m-1}, w_p) = \\ \lambda_1 P(c_m \mid c_p, c_{hp}, c_{m-1}, w_p) \\ + \lambda_2 P(c_m \mid c_p, c_{hp}, c_{m-1})$$

For part-of-speech tags, the mixture components are:
$$\tilde{P}(t_m \mid c_m, t_h, w_h) = \lambda_1 P(t_m \mid c_m, w_h) \\ + \lambda_2 P(t_m \mid c_m, t_h) \\ + \lambda_3 P(t_m \mid c_m)$$

For head words, the mixture components are:
$$\tilde{P}(w_m \mid c_m, t_m, t_h, w_h) = \lambda_1 P(w_m \mid c_m, t_m, w_h) \\ + \lambda_2 P(w_m \mid c_m, t_m, t_h) \\ + \lambda_3 P(w_m \mid c_m, t_m) \\ + \lambda_4 P(w_m \mid t_m)$$

Finally, for word features, the mixture components are:

$$P(f_m \mid c_m, t_m, t_h, w_h, known(w_m)) \approx$$
$$\lambda_1\ P(f_m \mid c_m, t_m, w_h, known(w_m))$$
$$\lambda_2\ P(f_m \mid c_m, t_m, t_h, known(w_m))$$
$$\lambda_3\ P(f_m \mid c_m, t_m, known(w_m))$$
$$\lambda_4\ P(f_m \mid t_m, known(w_m))$$

9   Searching the Model

Given a sentence to be analyzed, the search program must find the most likely semantic and syntactic interpretation. More precisely, it must find the most likely augmented parse tree. Although mathematically the model predicts tree elements in a top-down fashion, we search the space bottom-up using a chart-based search. The search is kept tractable through a combination of CKY-style dynamic programming and pruning of low probability elements.

9.1   Dynamic Programming

Whenever two or more constituents are equivalent relative to all possible later parsing decisions, we apply dynamic programming, keeping only the most likely constituent in the chart.   Two constituents are considered equivalent if:

1. They have identical category labels.

2. Their head constituents have identical labels.

3. They have the same head word.

4. Their leftmost modifiers have identical labels.

5. Their rightmost modifiers have identical labels.

9.2   Pruning

Given multiple constituents that cover identical spans in the chart, only those constituents with probabilities within a threshold of the highest scoring constituent are maintained; all others are pruned. For purposes of pruning, and only for purposes of pruning, the prior probability of each constituent category is multiplied by the generative probability of that constituent (Goodman, 1997). We can think of this prior probability as an estimate of the probability of generating a subtree with the constituent category, starting at the topmost node. Thus, the scores used in pruning can be considered as the product of:

1. The probability of generating a constituent of the specified category, starting at the topmost node.

2. The probability of generating the structure beneath that constituent, having already generated a constituent of that category.

Given a new sentence, the outcome of this search process is a tree structure that encodes both the syntactic and semantic structure of the sentence. The semantics – that is, the entities and relations – can then be directly extracted from these sentential trees.

10   Experimental Results

Our system for MUC-7 consisted of the sentential model described in this paper, coupled with a simple probability model for cross-sentence merging. The evaluation results are summarized in Table 1.

In both Template Entity (TE) and Template Relation (TR), our system finished in second place among all entrants. Nearly all of the work was done by the sentential model; disabling the cross-sentence model entirely reduced our overall F-Score by only 2 points.

Task	Recall	Precision	F-Score
Entities (TE)	83%	84%	83.49%
Relations (TR)	64%	81%	71.23%

Table 1: MUC-7 scores.

Task	Score
Part-of-Speech Tagging	95.99 (% correct)
Parsing (sentences < 40 words)	85.06 (F-Score)
Name Finding	92.28 (F-Score)

Table 2: Component task performance.

While our focus throughout the project was on TE and TR, we became curious about how well the model did at part-of-speech tagging, syntactic parsing, and at name finding. We evaluated part-of-speech tagging and parsing accuracy on the Wall Street Journal using a now standard procedure (see Collins 97), and evaluated name finding accuracy on the MUC-7 named entity test. The results are summarized in Table 2.

While performance did not quite match the best previously reported results for any of these three tasks, we were pleased to observe that the scores were at or near state-of-the-art levels for all cases.

11 Conclusions

We have demonstrated, at least for one problem, that a lexicalized, probabilistic context-free parser with head rules (LPCFG-HR) can be used effectively for information extraction. A single model proved capable of performing all necessary sentential processing, both syntactic and semantic. We were able to use the Penn TREEBANK to estimate the syntactic parameters; no additional syntactic training was required. The semantic training corpus was produced by students according to a simple set of guidelines. This simple semantic annotation was the only source of task knowledge used to configure the model.

Acknowledgements

References

Bikel, Dan; S. Miller; R. Schwartz; and R. Weischedel. (1997) "NYMBLE: A High-Performance Learning Name-finder." In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Association for Computational Linguistics, pp. 194-201.

Collins, Michael. (1996) "A New Statistical Parser Based on Bigram Lexical Dependencies." In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 184-191.

Collins, Michael. (1997) "Three Generative, Lexicalised Models for Statistical Parsing." In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 16-23.

Marcus, M.; B. Santorini; and M. Marcinkiewicz. (1993) "Building a Large Annotated Corpus of English: the Penn Treebank." *Computational Linguistics*, 19(2):313-330.

Goodman, Joshua. (1997) "Global Thresholding and Multiple-Pass Parsing." In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 11-25.

Placeway, P., R. Schwartz, et al. (1993). "The Estimation of Powerful Language Models from Small and Large Corpora." IEEE ICASSP

Weischedel, Ralph; Marie Meteer; Richard Schwartz; Lance Ramshaw; and Jeff Palmucci. (1993) "Coping with Ambiguity and Unknown Words through Probabilistic Models." *Computational Linguistics*, 19(2):359-382.