Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain, Philadelphia, July 2002, pp. 37-44. Association for Computational Linguistics.

A Transformational-based Learner for Dependency Grammars in Discharge Summaries

David A. Campbell M. Phil., Stephen B. Johnson Ph.D.
Department of Medical Informatics, Columbia University

Abstract

NLP systems will be more portable among medical domains if acquisition of semantic lexicons can be facilitated. We are pursuing lexical acquisition through the syntactic relationships of words in medical corpora. Therefore we require a syntactic parser which is flexible, portable, captures head-modifier pairs and does not require a large training set. We have designed a dependency grammar parser that learns through a transformational-based algorithm. We propose a novel design for templates and transformations which capitalize on the dependency structure directly and produces human-readable rules. Our parser achieved a 77% accurate parse training on only 830 sentences. Further work will evaluate the usefulness of this parse for lexical acquisition.

1 Introduction

Natural Language is a vital medium in medicine. Health care providers rely on medical narratives for recording, representing and sharing complex medical information such as the description of images, explanation of test results, or the summary of a patient's hospital visit. Natural Language Processing (NLP) tools have been applied to medical narrative for a variety of applications, such as triggering clinical alerts (Friedman, 1997) and document classification (Wilcox, 2000).

The effort required to create and maintain NLP systems in the medical setting can be prohibitive. Most language processors require a domain-specific semantic lexicon to function and, so far, these lexica have been created manually. The time and cost involved in creating these knowledge structures put limits on the extensibility and portability of NLP systems (Hripcsak, 1998). One solution to this bottleneck is to use machine learning to assist in categorizing lexemes into semantic classes. Such a tool could reduce the difficulty in porting NLP systems from one domain to another.

2 Dependency Grammars

One approach to semantic categorization is the use of syntactic features (Kokkinakis, 2001). This is based on the assumption that lexemes that share similar syntactic relations to other lexemes in the corpus will be semantically similar (Dorr, 2000). The idea of clustering words based on syntactic features has been well investigated in general language (Pereira, 1993; Li, 1998) However, (Harris, 1991) states that the syntactic relationships are more well-defined and have less variation in scientific languages (sublanguages), such as the ones used in medical texts. Identifying word classes using syntactic relationships should be simpler and potentially more useful in these types of languages.

Dependency grammars (Hudson, 1991) generate parses where words in a sentence are related directly to the word which is its syntactic head. Each word, except for the root has exactly one head, and the structure is a tree. The analysis does not generate any intermediate syntactic structures. Figure 1 shows an example of a sentence with a dependency grammar parse. There has been interest in learning dependency grammars from corpora. Collins (Collins, 1996) used dependencies as the backbone for his probabilistic parser and there has been work on learning both probabilistic (Carroll, 1992; Lee, 1999; Paskin, 2001) and transformation based dependency grammars (Hajic, 1997).

There are a number of attributes of dependency grammars which make them ideal for our goal of investigating medical sublanguage. First, the semantics of a word are often defined by a feature space of related words. The head-dependent relationships generated by a dependency parse can be used as the relationship for acquisition. Second, dependency grammars may be a better fit for parsing medical text. Medical text is frequently

include telegraphic omissions, run-on structures, improper use of conjunctions, left attaching noun modifiers etc (Sager, 1981). In many cases, many traditional phrase structures are absent or altered, making a phrase structure parse using traditional production rules difficult. A dependency grammar may still capture useful syntactic relationships when an accurate phrase grammar parse is not possible. In this way, a dependency parse may be compared to a shallow parse, in that it can return a partial analysis. However, even with a shallow parser, we would still interested in the dependency relationships inside the chunks. Third, the syntactic grammar of medical English, specifically regarding discharge summaries, is simpler overall (Campbell, 2001). We are not interested so much in the labeling of intermediate syntactic structures, such as noun phrases and prepositional phrases. Dependency grammars may allow us to capitalize on the relative syntactic simplicity of medical language without the overhead of generating and identifying structures which will not be used.

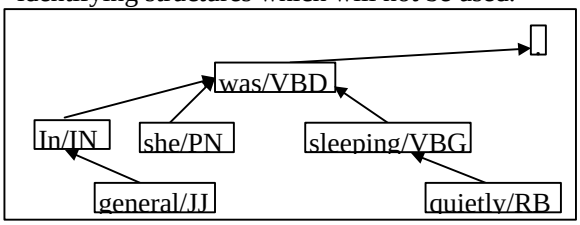


Figure 1. Dependency grammar parse of the sentence "In general she was sleeping quietly."

The dependency grammar used in this experiment did not allow crossing dependencies (projectivity). Crossing dependencies are ones where the parent and child of a relationship are on opposite sides of a common ancestor.

3 Transformational Based Learning

Transformational Based Learning (TBL) has been applied to numerous language learning problems, including part-of-speech tagging (Brill, 1994) , and parsing (Florian, 1998). It also has been used for learning dependency grammars (Hajic, 1997). In general, TBL algorithms generate smaller rule sets and require less training material than probabilistic approaches. Brill produced a part-of-speech tagger which was comparable in accuracy to other tagging methods.

In language, the general paradigm for TBL is to generate logical rules which apply transformations to the text. The training text is first annotated with the goal state. In this case, the sentences would be assigned a dependency parse. An initial state annotator is then applied to an unannotated copy of the text. For example, a right branching dependency tree was used in our experiment as the initial state (compare figure 1 and figure 2). The goal of TBL is to then generate rules which transform the naïve training state into the goal state. In order to do so, the TBL algorithm will have *templates* which describe the environment in the training corpus where a *transformation* can occur. The algorithm also has a *scoring function* which allows the comparison of the training state to the goal state. After iterating through the training corpus and testing all combinations of templates and transformations, the paired template and transformation which has the highest score becomes a *rule*. In other words, the best rule is the one which results in a corpus closest to the goal state after applying the transformation at the locations indicated by the template. This best rule is applied to the training corpus to produce a refined corpus. The process is then repeated, using the refined corpus as the training corpus, until no more positively scoring rules are produced. The final product is an ordered set of rules which can be applied to any unannotated corpus.

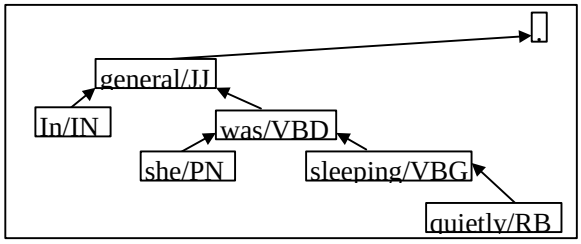


Figure 2. The initial dependency parse of the sentence "In general she was sleeping quietly."

TBL is a good choice for learning a dependency grammar of medical language. Assigning dependency heads is a task that is similar to part-of-speech tagging; each word in the text has exactly one dependency head, represented by the index of the head word. Transformations to this representation consist of

changing a word's dependency head from one word to another.

4 The Learning Algorithm

4.1 Template Design

In TBL, transformations occur when a specific environment in the text is found. These environments, or *triggers*, are defined by the proximal relationship of two or more parts of speech within a sentence. For example, in Brill's early work with POS tagging, one trigger was the existence of another specific POS tag immediately preceding the one to be transformed. The triggers, therefore, compose the 'if' component of the 'if-then' transformational rules.

When considering what triggers would be appropriate for dependency grammars, it was noted that many arcs in the grammar span a number of words. For example, the arc between a verb and the head of a noun phrase may span many words, especially in medical narratives where noun phrases can be especially lengthy. In previous attempts to parse language using TBL templates, the triggers have been tokens in the vicinity of the token to be transformed. While this has been successful for POS tagging, where the context necessary to correctly transform the tag may be found within two or three surrounding tokens, the distance of some dependency relationships can be much greater. In order to capture long distance relationships explicitly in a trigger, it would be necessary to expand the vicinity to be searched.

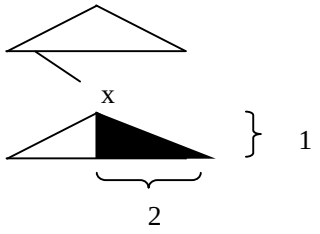
In the case of a dependency grammar parse, words are related to each other not only through their left-to-right arrangement, but also through the dependency tree. We sought to design triggers that take advantage of the dependency tree itself. Using the dependency relationships directly in the trigger is in the spirit of TBL where learning must change the triggering environments in the corpus from one iteration to the next. For example, in the case of POS tag learning, newly learned POS tags are used in subsequent iterations of the algorithm as triggers. Similarly, by using the dependency relationship directly in the trigger, we would expect the learner to capitalize on parse improvements through the learning process.

Each trigger used in this experiment had six parameters, which defined the vicinity around a target token, summarized in figure 3. Triggers can search using solely word distance, tree distance, or a combination of both. Any template can have multiple triggers, requiring multiple criteria to be met before considered true.

Trigger parameters

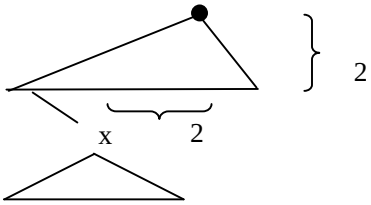
1. Word distance
2. Word direction (left, right, either)
3. Word scope (exactly at, within, all)
4. Tree distance
5. Tree direction (parent, child, either)
6. Tree scope (exactly at, within, all)

Example 1.



Trigger:
W-dist = 2 W-dir = right W-scp = within
T-dist = 1 T-dir = child T-scp = within

Example 2



Trigger:
W-dist = 2 W-dir = right W-scp = ex. at.
T-dist = 2 T-dir = par T-scp = ex. at

Figure 3. Trigger design and examples

The parameters of direction and distance are self-explanatory. Scope defines whether or not the triggering token must be *exactly at* the location defined by the distance, or *within* that distance. The third setting for scope is a special case. If the scope is set to *all* the template will search all tokens in the direction set, regardless of distance (e.g. if the tree direction is set to *left* and the scope is set to *all*, the trigger will match all tokens to the left, regardless of distance).

Two examples of triggers are given in figure 3. In both cases the triggers are searching for elements near token *x* which meet the correct criteria. In the first example, the trigger criteria will be met by any token within the shaded area of the tree, those tokens which are either one or two tokens to the right of *x* and are descendents of *x* with a tree distance of one. The second trigger will match a single token, shown as a black circle, that is exactly two tokens to the right of *x* and is also an ancestor of tree distance two.

4.2 Transformations

The second principal component of a TBL rule is the *transformation*, which defines a change to the structure of the sentence. For example, in the case of POS tagging, the transformation would be to change POS tag *x* to POS tag *y*. When TBL has been applied to parsing, the transformations have been on bracketed parse trees and have added or deleted brackets in a balanced method. Where the transformations seem intuitive for POS tagging, they are not as transparent for parsing. A rule for POS tagging may read, "If tag *x* is DT and tag *y* immediately to the right is VB, change tag *y* to NN." (see figure 4) This makes sense, for we do not expect verbs to immediately trail determiners, and transforming the verb to a noun would likely correct an error. A rule for parsing may read "If a bracket is immediately left of NN, delete a bracket to the left of the NN." This rule will combine a phrase which has a noun as the left-most component with the phrase which covers it. While this makes some sense, as many phrases do not have nouns as their left-most component, there are also many phrases which do. The linguistic motivation behind the transformation is not immediately obvious.

We wanted to give our transformations the intuitive readability of the rules seen in the POS tagging rules. In the case of our dependency grammar, we wanted our transformations to describe changes made directly to the tree. We considered four ways in which one token in the tree could be moved in relation to another outlined in figure 5. All four of the transformations decompose to the first transform. These transformations make intuitive sense for dependency grammars. We want to identify tokens in the text which are in the incorrect tree configuration and transform the tree by changing the dependency relationships. For example, the transformations "Make a noun the child of a verb" or "Make adjectives siblings of each other" are both readable in English and are linguistically reasonable.

Part-of-speech rule application

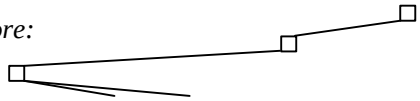
Before: The/**DT** fly/**VB** on/IN the/DT wall/NN

Apply Rule: If **VB** right of **DT** change **VB** to **NN**

After: The/**DT** fly/**NN** on/IN the/DT wall/NN

Bracketed-tree rule application

Before:



(((The/DT **(**fly/**NN** on/IN)) the/DT) wall/NN)

Apply Rule: If **'(**' left of **NN** delete **'(**' on left

After

(((The/DT fly/**NN** on/IN) the/DT) wall/NN)

Figure 4. Examples of applying transformations in POS tagging and parsing

Some transformations are disallowed in the special case that the root node is involved. The root node has no parent and can have no siblings and therefore transformations which would create these circumstances are not allowed. The shape of the dependency tree is restricted in other ways as described above, in that the trees have no crossing dependencies. These restrictions are not enforced by the transformations and it is possible that they could generate trees that violate these restrictions.

4.3 Rule Scoring

At every iteration, it is necessary to evaluate the goodness of the parse that results from the application of all tested rules. The rule which produces the best parse for that iteration is the one that is chosen and applied before continuing on to the next iteration. A number of measures for measuring parsing accuracy have been established, including bracketing sensitivity and specificity. Parsing accuracy for dependency

grammars is often measured as a function of the number of tokens which have the correct ancestors, or dependency accuracy. Keeping our goal of generating word-modifier pairs for subsequent machine learning, we chose an aggressive scoring function, counting only correct parent-child relationships. This also keeps the scoring function as simple as possible.

$$\text{Corpus score} = \frac{\text{\# correct dependencies}}{\text{\# of dependencies in corpus}}$$

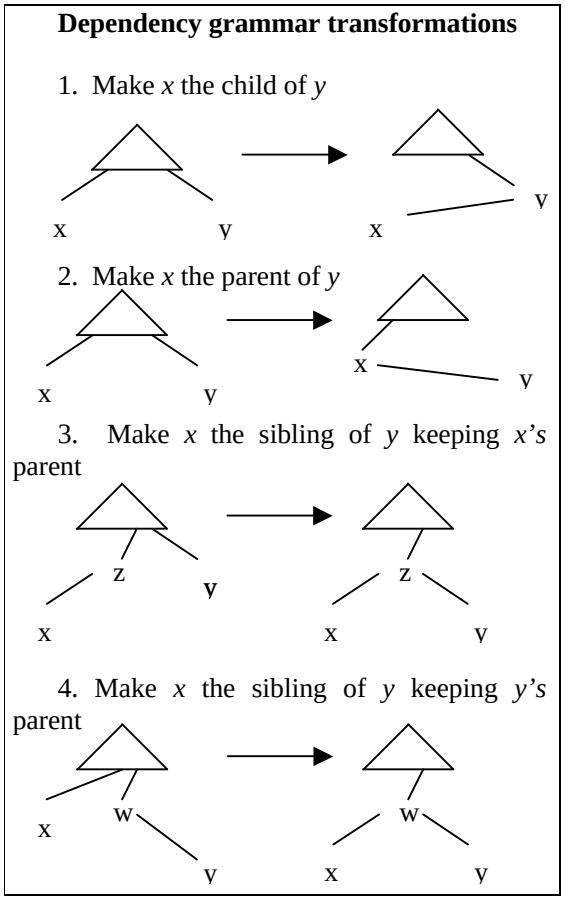


Figure 5. The four basic transformations

4.4 The Algorithm

The general design of TBL algorithms has been well described (Brill, 1994). The essential components, outlined above, include the template design, the transformations used, and the scoring system. The initial state of the dependency tree is the right branching tree shown in figure 2. To improve efficiency, we use the indexed TBL method outlined by Ramshaw and Marcus (Ramshaw, 1994). Rules have pointers to the sentences to which they apply, and similarly each sentence has pointers to the rules which have applied to it in the past. Rules are held on a heap based on their score, allowing the best rule to be found immediately after each iteration. The rule is applied to the list of sentences to which it points, and this list is used in the next iteration so no sentences which have not been modified need be seen.

5 Methods

A corpus of 1000 sentences (16,949 words) of text from medical discharge summaries was split into a training set of 830 sentences (13,954 words) and a test set of 170 sentences (2,995 words). The entire corpus was first POS tagged using a tagger trained specifically for discharge summaries (Campbell, 2001). The corpus was then hand parsed with a dependency grammar, and the TBL learner was allowed to learn rules on the training set. The sentences in the corpus were not restricted by length. Three sets of increasingly complex templates were used to learn rules, summarized in figure 6.

Template Set #1	
1. Word distance:	1, 2, or 3
2. Word direction:	left, right, or either
3. Word scope:	exactly at, within, or all
4. Tree distance:	not used
5. Tree direction:	not used
6. Tree scope:	not used
Template Set #2	**all of set 1, and . . .**
1. Word distance:	not used
2. Word direction:	not used
3. Word scope:	not used
4. Tree distance:	1, 2, or 3
5. Tree direction:	child, parent or either
6. Tree scope:	exactly at, within, or all
Template Set #3	**all of set 1, 2, and. . .**
1. Word distance:	1,2 or 3
2. Word direction:	left, right, or either
3. Word scope:	exactly at, within, or all
4. Tree distance:	1, 2, or 3
5. Tree direction:	child, parent or either
6. Tree scope:	exactly at, within, or all

Figure 6. Three template sets used

6 Results

The three template sets generated three rule sets, each of which was evaluated on the 170 sentence test set. Each template set was trained with increasing amounts of the training corpus to measure the effect of the training set size on the learner's accuracy. Chart 1 shows the improvement in accuracy gained through larger training sets. The best dependency accuracy and number of rules generated for each template set is reported in table 1.

Template Set	Rules tested	Parser Rules	Tree Rules	Parse acc.
Set 1	48K+	424	0	76.5%
Set 2	93K+	498	127	77.0%
Set 3	187K+	541	249	77.0%

Table 1. Results for three template sets used

Test set	Sentence length	Total Sents.	Avg. Length	Parse acc.
1	n <= 10	61	7.1	87.6%
2	n <= 20	127	11.4	80.1%
Full	all n	200	17.9	77.0%

Table 2. Effect of sentence length on accuracy

To measure the effect of sentence length on parsing accuracy, the best parser rules were re-tested on two subsets of the test sets. The first subset contained sentences with a length less than ten words and the second contained sentences of length less than twenty. The resulting accuracy of the parser on these sentences is summarized in table 2. The top ten rules acquired with the third template set are reported in table 3.

7 Discussion and Further Work

For all sets of templates, the learner produced a rule-based parser with dependency accuracy exceeding 75% when sentence length was not restricted. For the best parser generated, limiting the sentence length to 20 and 10 words improved the parsing accuracy to 80.1% and 87.6%. Little difference among the template sets was found, although the use of tree-based templates gave slightly better performance. Although we expected the inclusion of tree based templates to improve the performance of the parser by a greater extent than observed, it is significant that the learner was reasonably successful with only word-order information. The strongest syntactic dependencies in medical language may be local and the addition of the tree-oriented templates is not very significant. However, when the tree information is available, the learner does use it, as can be seen by the number of rules using tree information in the three sets shows (table 1). For the third template set, 46% of the rules learned incorporated tree information.

	Base POS	Trg. POS	W Dis	W Dir	W Scp	T Dis	T Dir	T Scp	Xf
1	NN	IN	3	-1	2	0	0	0	2
2	IN	NN	-1	1	2	0	0	0	2
3	VBD	NN	0	0	0	2	1	1	1
4	JJ	NN	2	1	2	0	0	0	2
5	VBD	VBN	3	1	2	0	0	0	1

Base POS	{POS of current token}
Trig. POS	{POS to be found by trigger}
W/T Dis	{# of tokens trigger is from base}
W Dir	{1 = right, -1 = left, 0 = either }
T Dir	{1 = par, -1 = child, 0 = either }
W/T Scp	{1 = at, 2 = within, 3 = all}
Xf	{1 = make child, 2 = make par}

Table 3. First 5 rules learned by template set 3

The rules generated are easily translated into English and make good linguistic sense. The first rule in Table 2 reads "If this is a singular noun (Base POS = NN) and there is a preposition (Trg POS = IN) within (W. scp = 2) three tokens (W. dis = 3) to the left (W. dir =-1) then make the preposition the parent of the noun." This is the type of rule we would expect to see, as it begins forming prepositional phrases attaching to prepositions on the left. The third rule uses information in the dependency tree, reading "If this is a simple past verb and there is a singular noun that is the grandparent, make that noun the child of the verb."

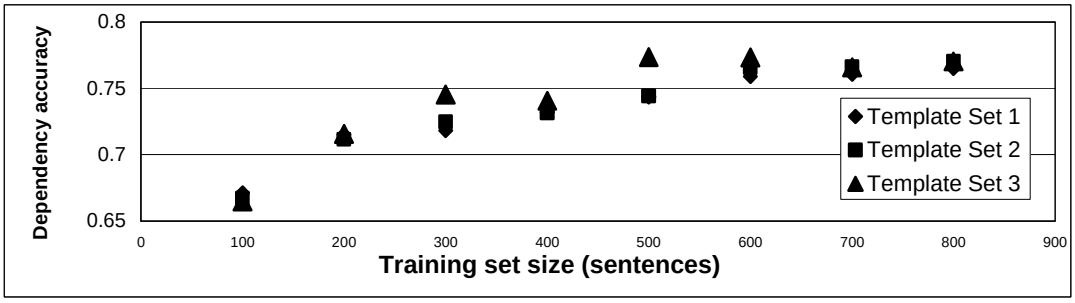


Chart 1. Effect of training set size on dependency accuracy for three template sets

The greatest drawback to this approach is the computing requirements. The consequence of the complex template design used is a large number of rules which need to be kept in memory. The third template set generated over 187,000 rules which need to be stored in memory. Of these, only 240 rules were kept in the rule set. Because each rule needs to store a list of pointers back to the sentences to which it applied, the size of a rule grows with the size of the training set. It will be crucial to incorporate rule pruning in the future to allow larger training sets and more complex templates.

Although the results shown here are for training on a specific corpus of discharge summaries, the learning algorithm itself is domain independent. We foresee generating parsers on a number of medical corpora, including radiology reports, pathology reports and progress notes. Therefore, we require a flexible solution that would not demand reengineering the parser for every new domain. The learning algorithm described here could be used on any general corpus where the sentences can be given a dependency parse. We intend to evaluate the algorithm on more general corpora in the future.

Overall, the results are very encouraging. Keeping in mind our goal of gathering head-modifier pairs for machine learning, a 77% accurate parse is approaching an acceptable parse (Sekine, 1992). The results also show that limiting the sentence length can improve the accuracy of the parser. If our sole desire is the generation of head-modifier pairs, using a large number of shorter sentences may be equivalent to using fewer longer ones. We also believe that the parser may be improved through lexicalization, but that remains future work.

The ability to generate a good parser from such a small training set is important in the medical domain. Previous work has shown that different medical domains have to be treated as separate languages for successful NLP (Friedman, 1995). Therefore, it is likely that any medical domain we wish to parse will require its own training set for the parser. If extensive training set preparation was required, then we are simply trading one difficult task for another: the task of manually creating and maintaining a semantic lexicon with the task of hand dependency-parsing large amounts of text. Although the task of hand-parsing 1,000 sentences of discharge summaries is not trivial, it is reasonable and manageable and does not require extensive medical knowledge. The shift-reduce parser described by (Hermjakob, 1997) also requires relatively few training examples but requires semantic features that may require medical knowledge to construct and assign. Although we do not propose here a specific application for a dependency grammar in a medical domain, we believe it will be valuable for future clustering, disambiguation and indexing applications.

8 Conclusions

Natural language processors in the medical domain will be more flexible and portable with assisted lexicon design. The syntactic dependencies in a dependency grammar may be useful for the lexical acquisition necessary to make this possible. We have investigated using transformational-based learning as a technique for learning a dependency grammar in a medical corpus. To better learn dependency grammars we used a template design which uses the structure of the parse tree explicitly and transformations that operated directly on the trees. Training on a set of 830 sentences of parsed medical discharge summaries gave a best

parser with 77% accuracy. The inclusion of tree information in the template design slightly improved the parser. The rules produced were intuitive and understandable, and the limited amount of training material will allow the technique to be used on other medical domains without extensive manual parsing. Further work will test the utility of head-dependency relationships for machine learning semantic classes.

References

Brill, E. (1994). A report of recent progress in transformation-based error-driven learning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Princeton, NJ.

Campbell, D.A., Johnson, S.B. (2001). Comparing Syntactic Complexity in Medical and non-Medical Corpora. In *Proc AMIA Annu Fall Symp.* 90-94.

Carroll, G., Charniak, E. (1992). Two experiments on learning probabilistic dependency grammars from corpora. In *Workshop Notes for Statistically-Based NLP Techniques*, AAAI. 1-13.

Collins, M. (1996.). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of ACL*. 184-191.

Dorr, B. J., Jones, D. (2000). Acquisition of Semantic Lexicons: Using Word Sense Disambiguation to Improve Precision, in Evelyn Viegas (Ed), *Breadth and Depth of Semantic Lexicons*, Kluwer Academic Publishers: Norwell, MA, 79-98.

Florian, R., Brill, E. (1998). Transformation Based Parsing. *Ph.D. Qualifier Project*, Computer Science Department, Johns Hopkins University.

Friedman, C. (1997). Towards a comprehensive medical language processing system: methods and issues. In *Proc AMIA Annu Fall Symp*. 595-9.

Friedman C, et. al (1995). Architectural requirements for a multipurpose natural language processor in the clinical environment. In *Proc 19th Annu SCAMC*. 347-51.

Hajic, J. and K. Ribarov (1997). Rule-Based Dependencies. *Workshop on Empirical Learning of Natural Language Processing Tasks*, Prague, Czech Republic.

Harris, Z. (1991). *A Theory of Language and Information*. Oxford University Press: Oxford.

Hermjakob U. & Mooney R.J. (1997) Learning Parse and Translation Decisions From Examples With Rich Context, *Proc. of ACL-EACL Conf.* 482-489.

Hripcsak, G., G. Kuperman, et al. (1998). Extracting Findings from Narrative Reports: Software Transferability and Sources of Physician Disagreement. *Methods Inf Med* 37. 1-7.

Hudson, Richard. (1991). *English Word Grammar*. Blackwell: Cambridge, Mass.

Kokkinakis D. (2001), Syntactic Parsing as a Step for Automatically Augmenting Semantic Lexicons, In *Proceedings of the 39th ACL and the 10th EACL*, Toulouse, France. Student Workshop. 13-18.

Lee, S. and K.-S. Choi (1999). A Reestimation Algorithm for Probabilistic Dependency Grammars. *Natural Language Engineering* 5(3). 251-270.

Li H. and Abe N.(1998). Word clustering and disambiguation based on co-occurrence data. In Proceedings of COLING - ACL'98. 749-755

Paskin, M. (2001). Grammatical Bigrams. In T. Dietterich, S. Becker, and Z. Gharahmani eds., *Advances in Neural Information Processing Systems 14*. MIT Press: Cambridge, MA.

Pereira F., Tishby, N., Lee, L (1993). Distributional clustering of English words. In *30th Annual Meeting of the ACL*, 183-190.

Ramshaw, L. A., Marcus, M. P. (1994). Exploring the statistical derivation of transformational rule sequences for part-of-speech tagging. In *Proceedings of the Balancing Act Workshop on Combining Symbolic and Statistical Approaches to Language*, Association for Computational Linguistics. 86-95.

Sager N. (1981). *Natural Language Information Processing: A Computer Grammar of English and its Applications*. Addison-Wesley: Reading, Massachusetts.

Sekine, S. et. al (1992) Automatic Learning for Semantic Collocation. In *3rd Conf. on Applied Natural Language Processing* :Trent - Italy.

Wilcox, A, Hripcsak G. (2000). Medical Text Representations for Inductive Learning. *Proc AMIA Symp*. 923-7.