A Comparison of Alternative Parse Tree Paths for Labeling Semantic Roles

Reid Swanson and Andrew S. Gordon
Institute for Creative Technologies
University of Southern California
13274 Fiji Way, Marina del Rey, CA 90292 USA
swansonr@ict.usc.edu, gordon@ict.usc.edu

Abstract

The integration of sophisticated inference-based techniques into natural language processing applications first requires a reliable method of encoding the predicate-argument structure of the propositional content of text. Recent statistical approaches to automated predicate-argument annotation have utilized parse tree paths as predictive features, which encode the path between a verb predicate and a node in the parse tree that governs its argument. In this paper, we explore a number of alternatives for how these parse tree paths are encoded, focusing on the difference between automatically generated constituency parses and dependency parses. After describing five alternatives for encoding parse tree paths, we investigate how well each can be aligned with the argument substrings in annotated text corpora, their relative precision and recall performance, and their comparative learning curves. Results indicate that constituency parsers produce parse tree paths that can more easily be aligned to argument substrings, perform better in precision and recall, and have more favorable learning curves than those produced by a dependency parser.

1 Introduction

A persistent goal of natural language processing research has been the automated transformation of natural language texts into representations that unambiguously encode their propositional content in formal notation. Increasingly, first-order predicate calculus representations of textual meaning have been used in natural lanugage processing applications that involve automated inference. For example, Moldovan et al. (2003) demonstrate how predicate-argument formulations of questions and candidate answer sentences are unified using logical inference in a top-performing question-answering application. The importance of robust techniques for predicate-argument transformation has motivated the development of large-scale text corpora with predicate-argument annotations such as PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998). These corpora typically take a pragmatic approach to the predicate-argument representations of sentences, where predicates correspond to single word triggers in the surface form of the sentence (typically verb lemmas), and arguments can be identified as substrings of the sentence.

Along with the development of annotated corpora, researchers have developed new techniques for automatically identifying the arguments of predications by labeling text segments in sentences with semantic roles. Both Gildea & Jurafsky (2002) and Palmer et al. (2005) describe statistical labeling algorithms that achieve high accuracy in assigning semantic role labels to appropropriate constituents in a parse tree of a sentence. Each of these efforts employed the use of *parse tree paths* as predictive features, encoding the series of up and down transitions through a parse tree to move from the node of the verb (predicate) to the governing node of the constituent (argument). Palmer et al. (2005) demonstrate that utilizing the gold-standard parse trees of the Penn treebank (Marcus et al., 1993) to encode parse tree paths yields significantly better labeling accuracy than when using an automatic syntactical parser, namely that of Collins (1999).

Parse tree paths (between verbs and arguments that fill semantic roles) are particularly interesting because they symbolically encode the relationship between the syntactic and semantic aspects of verbs, and are potentially generalized across other verbs within the same class (Levin, 1993). However, the encoding of individual parse tree paths for predicates is wholly dependent on the characteristics of the parse tree of a sentence, for which competing approaches could be taken.

The research effort described in this paper further explores the role of parse tree paths in identifying the argument structure of verb-based predications. We are particularly interested in exploring alternatives to the constituency parses that were used in previous research, including parsing approaches that employ dependency grammars. Specifically, our aim is to answer four important questions:

1. How can parse tree paths be encoded when employing different automated constituency parsers, i.e. Charniak (2000), Klein & Manning (2003), or a dependency parser (Lin, 1998)?

2. Given that each of these alternatives creates a different formulation of the parse tree of a sentence, which of them encodes branches that are easiest to align with substrings that have been annotated with semantic role information?

3. What is the relative precision and recall performance of parse tree paths formulated using these alternative automated parsing techniques, and do the results vary depending on argument type?

4. How many examples of parse tree paths are necessary to provide as training examples in order to achieve high labeling accuracy when employing each of these parsing alternatives?

Each of these four questions is addressed in the four subsequent sections of this paper, followed by a discussion of the implications of our findings and directions for future work.

2 Alternative Parse Tree Paths

Parse tree paths were introduced by Gildea & Jurafsky (2002) as descriptive features of the syntactic relationship between predicates and arguments in the parse tree of a sentence. Predicates are typically assumed to be specific target words (usually verbs), and arguments are assumed to be a span of words in the sentence that are governed by a single node in the parse tree. A parse tree path can be described as a sequence of transitions up and down a parse tree from the target word to the governing node, as exemplified in Figure 1.

The encoding of the parse tree path feature is dependent on the syntactic representation that is produced by the parser. This, in turn, is dependant on the training corpus used to build the parser, and the conditioning factors in its probability model. As result, encodings of parse tree paths can vary greatly depending on the parser that is used, yielding parse tree paths that vary in their ability to generalize across sentences.

In this paper we explore the characteristics of parse tree paths with respect to different approaches to automated parsing. We were particularly interested in comparing traditional constituency parsing (as exemplified in Figure 1) with dependency parsing, specifically the Minipar system built by Lin (1998). Minipar is increasingly being used in semantics-based nlp applications (e.g. Pantel & Lin, 2002). Dependency parse trees differ from constituency parses in that they represent sentence structures as a set of dependency relationships between words, typed asymmetric binary relationships between head words and modifying words. Figure 2 depicts the output of Minipar on an example sentence, where each node is a word or an empty node along with the word lemma, its part of speech, and the relationship type to its governing node.

Our motivation for exploring the use of Minipar in for the creation of parse tree paths can be seen by comparing Figure 1 and Figure 2, where

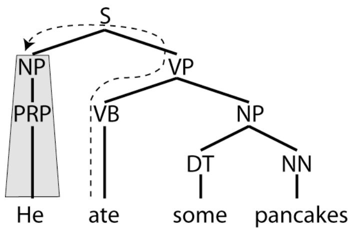

Figure 1: An example parse tree path from the predicate *ate* to the argument NP *He*, represented as VB↑VP↑S↓NP.

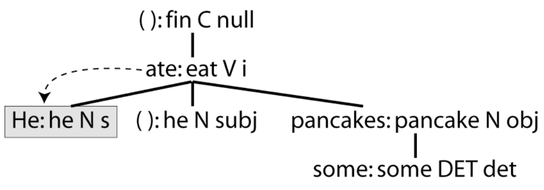

Figure 2. An example dependency parse, with a parse tree path from the predicate *ate* to the argument *He*.

the Minipar path is both shorter and simpler for the same predicate-argument relationship, and could be encoded in various ways that take advantage of the additional semantic and lexical information that is provided.

To compare traditional constituency parsing with dependency parsing, we evaluated the accuracy of argument labeling using parse tree paths generated by two leading constituency parsers and three variations of parse tree paths generated by Minipar, as follows:

Charniak: We used the Charniak parser (2000) to extract parse tree paths similar to those found in Palmer et al. (2005), with some slight modifications. In cases where the last node in the path was a non-branching pre-terminal, we added the lexical information to the path node. In addition, our paths led to the lowest governing node, rather than the highest. For example, the parse tree path for the argument in Figure 1 would be encoded as:

VB↑VP↑S↓NP↓PRP:he

Stanford: We also used the Stanford parser developed by Klein & Manning (2003), with the same path encoding as the Charniak parser.

Minipar A: We used three variations of parse tree path encodings based on Lin's dependency parser, Minipar (1998). Minipar A is the first and most restrictive path encoding, where each is annotated with the entire information output by Minpar at each node. A typical path might be:

ate:eat,V,i↓He:he,N,s

Minipar B: A second parse tree path encoding was generated from Minipar parses that relaxes some of the constraints used in Minpar A. Instead of using all the information contained at a node, in Minipar B we only encode a path with its part of speech and relational information. For example:

V,i↓N,s

Minipar C: As the converse to Minipar A we also tried one other Minipar encoding. As in Minipar A, we annotated the path with all the information output, but instead of doing a direct string comparison during our search, we considered two paths matching when there was a match between either the word, the stem, the part of speech, or the relation. For example, the following two parse tree paths would be considered a match, as both include the relation *i*.

ate:eat,V,i↓He:he,N,s
was:be,VBE,i↓He:he,N,s

We explored other combinations of dependency relation information for Minipar-derived parse tree paths, including the use of the deep relations. However, results obtained using these other combinations were not notably different from those of the three base cases listed above, and are not included in the evaluation results reported in this paper.

3 Aligning arguments to parse trees nodes in a training / testing corpus

We began our investigation by creating a training and testing corpus of 400 sentences each containing an inflection of one of four target verbs (100 each), namely *believe*, *think*, *give*, and *receive*. These sentences were selected at random from the 1994-07 section of the New York Times gigaword corpus from the Linguistic Data Consortium. These four verbs were chosen because of the synonymy among the first two, and the reflexivity of the second two, and because all four have straightforward argument structures when viewed as predicates, as follows:

predicate: **believe**
arg0: the believer
arg1: the thing that is believed

predicate: **think**
arg0: the thinker
arg1: the thing that is thought

predicate: **give**
arg0: the giver
arg1: the thing that is given
arg2: the receiver

predicate: **receive**
arg0: the receiver
arg1: the thing that is received
arg2: the giver

This corpus of sentences was then annotated with semantic role information by the authors of this paper. All annotations were made by assigning start and stop locations for each argument in the unparsed text of the sentence. After an initial pilot annotation study, the following annotation policy was adopted to overcome common disagreements: (1) When the argument is a noun and it is part of a definite description then include the entire definite description. (2) Do not include complementizers such as ‘that’ in ‘believe that’ in an argument. (3) Do include prepositions such as ‘in’ in ‘believe in’. (4) When in doubt, assume phrases attach locally. Using this policy, an agreement of 92.8% was achieved among annotators for the set of start and stop locations for arguments. Examples of semantic role annotations in our corpus for each of the four predicates are as follows:

1. [$_{\text{Arg0}}$Those who excavated the site in 1907] believe [$_{\text{Arg1}}$ it once stood two or three stories high.]

2. Gus is in good shape and [$_{\text{Arg0}}$ I] think [$_{\text{Arg1}}$ he's happy as a bear.]

3. If successful, [$_{\text{Arg0}}$ he] will give [$_{\text{Arg1}}$ the funds] to [$_{\text{Arg2}}$ his Vietnamese family.]

4. [$_{\text{Arg0}}$ The Bosnian Serbs] have received [$_{\text{Arg1}}$ military and economic support] from [$_{\text{Arg2}}$ Serbia.]

The next step was to parse the corpus of 400 sentences using each of three automated parsing systems (Charniak, Stanford, and Minipar), and align each of the annotated arguments with its closest matching branch in the resulting parse trees. Given the differences in the parsing models used by these three systems, each yield parse tree nodes that govern different spans of text in the sentence. Often there exists no parse tree node that governs a span of text that exactly matches the span of an argument in the annotated corpus. Accordingly, it was necessary to identify the closest match possible for each of the three parsing systems in order to encode parse tree paths for each. We developed a uniform policy that would facilitate a fair comparison between parsing techniques. Our approach was to identify a single node in a given parse tree that governed a string of text with the most overlap with the text of the annotated argument. Each of the parsing methods tokenizes the input string differently, so in order to simplify the selection of the governing node with the most overlap, we made this selection based on lowest minimum edit distance (Levenshtein distance).

All three of these different parsing algorithms produced single governing nodes that overlapped well with the human-annotated corpus. However, it appeared that the two constituency parsers produced governing nodes that were more closely aligned, based on minimum edit distance. The Charniak parser aligned best with the annotated text, with an average of 2.40 characters for the lowest minimum edit distance (standard deviation = 8.64). The Stanford parser performed slightly worse (average = 2.67, standard deviation = 8.86), while distances were nearly two times larger for Minipar (average = 4.73, standard deviation = 10.44).

In each case, the most overlapping parse tree node was treated as correct for training and testing purposes.

4 Comparative Performance Evaluation

In order to evaluate the comparative performance of the parse tree paths for each of the five encodings, we divided the corpus in to equal-sized training and test sets (50 training and 50 test examples for each of the four predicates). We then constructed a system that identified the parse tree paths for each of the 10 arguments in the training sets, and applied them to the sentences in each corresponding test sets. When applying the 50 training parse tree paths to any one of the 50 test sentences for a given predicate-argument pair, a set of zero or more candidate answer nodes were returned. For the purpose of calculating precision and recall scores, credit was given when the correct answer appeared in this set. Precision scores were calculated as the number of correct answers found divided by the number of all candidate answer nodes returned. Recall scores were calculated as the number of correct answers found divided by the total number of correct answers possible. F-scores were calculated as the equally-weighted harmonic mean of precision and recall.

Our calculation of recall scores represents the best-possible performance of systems using only these types of parse-tree paths. This level of performance could be obtained if a system could always select the correct answer from the set of candidates returned. However, it is also informative to estimate the performance that could be achieved by randomly selecting among the candidate answers, representing a lower-bound on performance. Accordingly, we computed an adjusted recall score that awarded only fractional credit in cases where more than one candidate answer was returned (one divided by the set size). Adjusted recall is the sum of all of these adjusted credits divided by the total number of correct answers possible.

Figure 3 summarizes the comparative recall, precision, f-score, and adjusted recall performance for each of the five parse tree path formulations. The Charniak parser achieved the highest overall scores (precision=.49, recall=.68, f-score=.57, adjusted recall=.48), followed closely

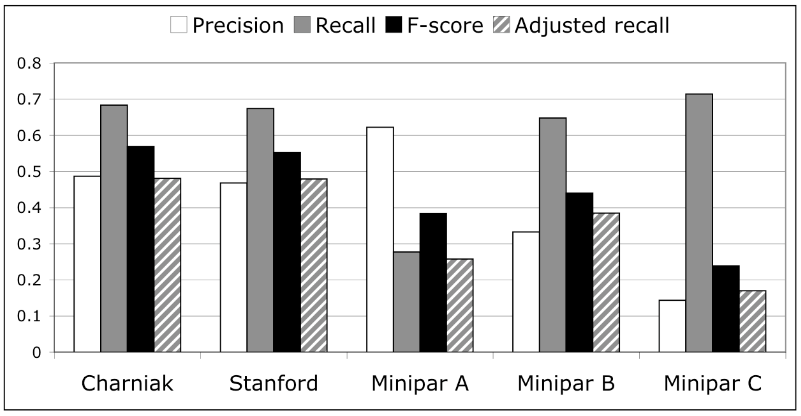

Figure 3. Precision, recall, f-scores, and adjusted recall for five parse tree path types

by the Stanford parser (precision=.47, recall=.67, f-score=.55, adjusted recall=.48).

Our expectation was that the short, semantically descriptive parse tree paths produced by Minipar would yield the highest performance. However, these results indicate the opposite; the constituency parsers produce the most accurate parse tree paths. Only Minipar C offers better recall (0.71) than the constituency parsers, but at the expense of extremely low precision. Minipar A offers excellent precision (0.62), but with extremely low recall. Minipar B provides a balance between recall and precision performance, but falls short of being competitive with the parse tree paths generated by the two constituency parsers, with an f-score of .44.

We utilized the Sign Test in order to determine the statistical significance of these differences. Rank orderings between pairs of systems were determined based on the adjusted credit that each system achieved for each test sentence. Significant differences were found between the performance of every system ($p<0.05$), with the exception of the Charniak and Stanford parsers. Interestingly, by comparing weighted values for each test example, Minipar C more frequently scores higher than Minipar A, even though the sum of these scores favors Minipar A.

In addition to overall performance, we were interested in determining whether performance varied depending on the type of the argument that is being labeled. In assigning labels to arguments in the corpus, we followed the general principles set out by Palmer et al. (2005) for labeling arguments arg0, arg1 and arg2. Across each of our four predicates, arg0 is the agent of the predication (e.g. the person that has the belief or is doing the giving), and arg1 is the thing that is acted upon by the agent (e.g. the thing that is believed or the thing that is given). Arg2 is used only for the predications based on the verbs *give* and *receive*, where it is used to indicate the other party of the action.

Our interest was in determining whether these five approaches yielded different results depending on the semantic type of the argument. Figure 4 presents the f-scores for each of these encodings across each argument type.

Results indicate that the Charniak and Stanford parsers continue to produce parse tree paths that outperform each of the Minipar-based approaches. In each approach argument 0 is the easiest to identify. Minipar A retains the general trends of Charniak and Stanford, with argument

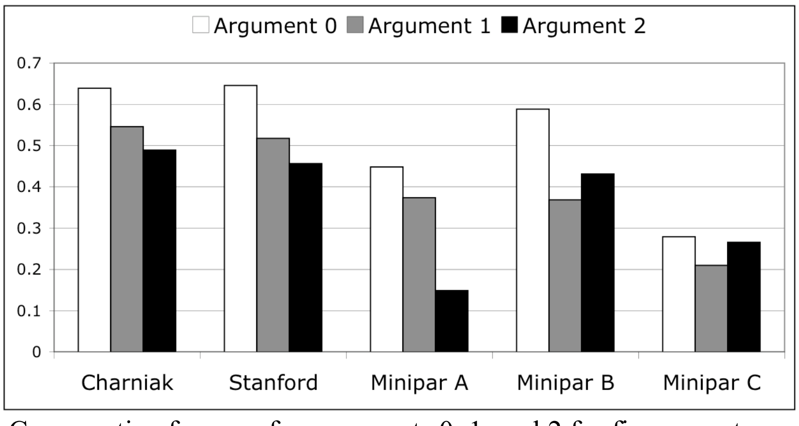

Figure 4. Comparative f-scores for arguments 0, 1, and 2 for five parse tree path types

1 easier to identify than argument 2, while Minipar B and C show the reverse. The highest f-scores for argument 0 were achieved Stanford (f=.65), while Charniak achieved the highest scores for argument 1 (f=.55) and argument 2 (f=.49).

5 Learning Curve Comparisons

The creation of large-scale text corpora with syntactic and/or semantic annotations is difficult, expensive, and time consuming. The PropBank effort has shown that producing this type of corpora is considerably easier once syntactic analysis has been done, but substantial effort and resources are still required. Better estimates of total costs could be made if it was known exactly how many annotations are necessary to achieve acceptable levels of performance. Accordingly, we investigated the learning curves of precision, recall, f-score, and adjusted recall achieved using the five different parse tree path encodings.

For each encoding approach, learning curves were created by applying successively larger subsets of the training parse tree paths to each of the items in the corresponding test set. Precision, recall, f-scores, and adjusted recall were computed as described in the previous section, and identical subsets of sentences were used across parsers, in one-sentence increments. Individual learning curves for each of the five approaches are given in Figures 5, 6, 7, 8, and 9. Figure 10 presents a comparison of the f-score learning curves for all five of the approaches.

In each approach, the precision scores slowly degrade as more training examples are provided, due to the addition of new parse tree paths that yield additional candidate answers. Conversely, the recall scores of each system show their greatest gains early, and then slowly improve with the addition of more parse tree paths. In each approach, the recall scores (estimating best-case performance) have the same general shape as the adjusted recall scores (estimating the lower-bound performance). The divergence between these two scores increases with the addition of more training examples, and is more pronounced in systems employing parse tree paths with less specific node information. The comparative f-score curves presented in Figure 10 indicate that Minipar B is competitive with Charniak and Stanford when only a small number of training examples is available. There is some evidence here that the performance of Minipar A would continue to improve with the addition of more training data, suggesting that this approach might be well-suited for applications where lots of training data is available.

6 Discussion

Annotated corpora of linguistic phenomena enable many new natural language processing applications and provide new means for tackling difficult research problems. Just as the Penn Treebank offers the possibility of developing systems capable of accurate syntactic parsing, corpora of semantic role annotations open up new possibilities for rich textual understanding and integrated inference.

In this paper, we compared five encodings of parse tree paths based on two constituency parsers and a dependency parser. Despite our expectations that the semantic richness of dependency parses would yield paths that outperformed the others, we discovered that parse tree paths from Charniak's constituency parser performed the best overall. In applications where either precision or recall is the only concern, then Minipar-derived parse tree paths would yield the best results. We also found that the performance of all of these systems varied across different argument types.

In contrast to the performance results reported by Palmer et al. (2005) and Gildea & Jurafsky (2002), our evaluation was based solely on parse tree path features. Even so, we were able to obtain reasonable levels of performance without the use of additional features or stochastic methods. Learning curves indicate that the greatest gains in performance can be garnered from the first 10 or so training examples. This result has implications for the development of large-scale corpora of semantically annotated text. Developers should distribute their effort in order to maximize the number of predicate-argument pairs with at least 10 annotations.

An automated semantic role labeling system could be constructed using only the parse tree path features described in this paper, with estimated performance between our recall scores and our adjusted recall scores. There are several ways to improve on the random selection approach used in the adjusted recall calculation. For example, one could simply select the candidate answer with the most frequent parse tree path.

The results presented in this paper help inform the design of future automated semantic role labeling systems that improve on the best-performing systems available today (Gildea &

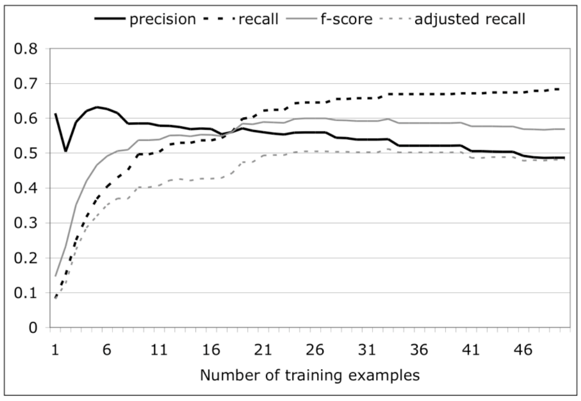

Figure 5. Charniak learning curves

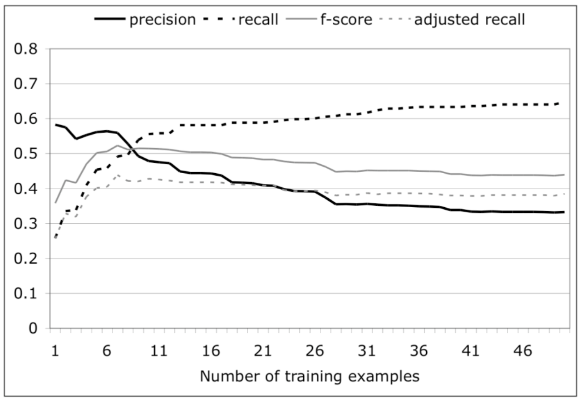

Figure 8. Minipar B learning curves

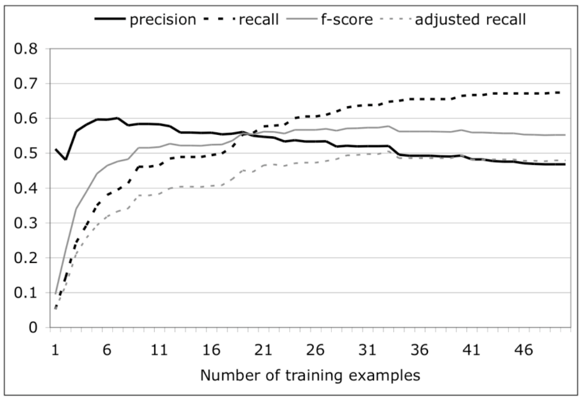

Figure 6. Stanford learning curves

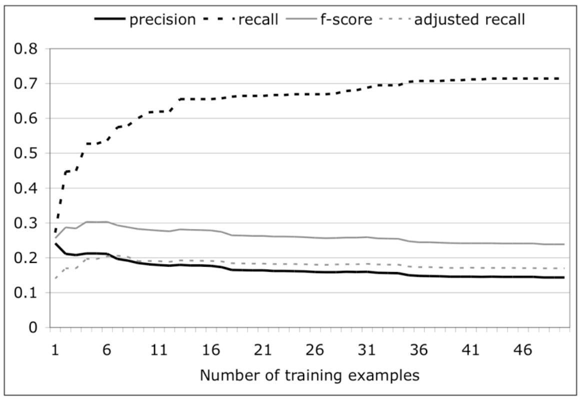

Figure 9. Minipar C learning curves

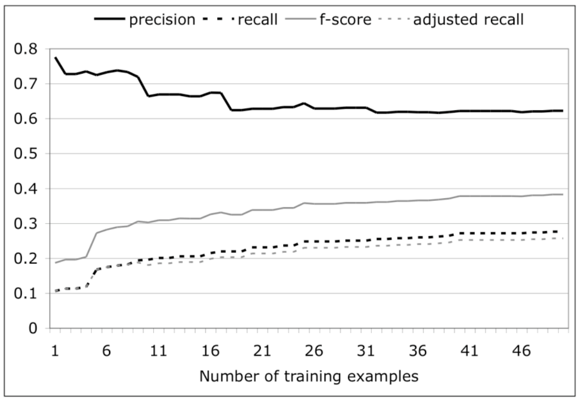

Figure 7. Minipar A learning curves

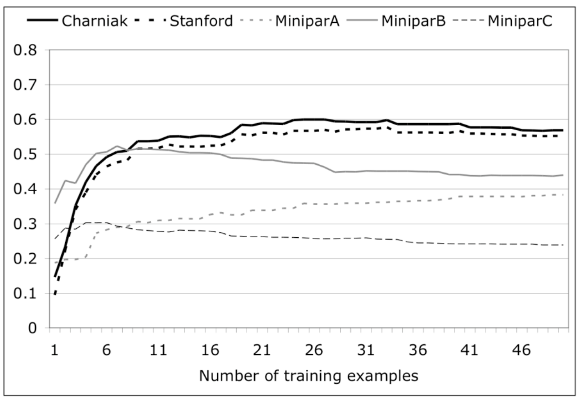

Figure 10. Comparative F-score curves

Jurafsky, 2002; Moschitti et al., 2005). We found that different parse tree paths encode different types of linguistic information, and exhibit different characteristics in the tradeoff between precision and recall. The best approaches in future systems will intelligently capitalize on these differences in the face of varying amounts of training data.

In our own future work, we are particularly interested in exploring the regularities that exist among parse tree paths for different predicates. By identifying these regularities, we believe that we will be able to significantly reduce the total number of annotations necessary to develop lexical resources that have broad coverage over natural language.

Acknowledgments

The project or effort depicted was sponsored by the U. S. Army Research, Development, and Engineering Command (RDECOM). The content or information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

Baker, Collin, Charles J. Fillmore and John B. Lowe. 1998. The Berkeley FrameNet Project, In Proceedings of COLING-ACL, Montreal.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser, In Proceedings NAACL.

Collins, Michael. 1999. Head-Driven Statistical Models for Natural Language Parsing, PhD thesis, University of Pennsylvania.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles, Computational Linguistics, 28(3):245--288.

Klein, Dan and Christopher Manning. 2003. Accurate Unlexicalized Parsing, In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 423-430.

Levin, Beth. 1993. English Verb Classes and Alternations: A Preliminary Investigation. Chicago, IL: University of Chicago Press.

Lin, Dekang. 1998. Dependency-Based Evaluation of MINIPAR, In Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation, Granada, Spain.

Marcus, Mitchell P., Beatrice Santorini and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank, Computational Linguistics, 19(35):313-330

Moldovan, Dan I., Christine Clark, Sanda M. Harabagiu & Steven J. Maiorano. 2003. COGEX: A Logic Prover for Question Answering, HLT-NAACL.

Moschitti, A., Giuglea, A., Coppola, B. & Basili, R. 2005. Hierarchical Semantic Role Labeling. In Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL 2005 shared task), Ann Arbor(MI), USA.

Palmer, Martha, Dan Gildea and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles, Computational Linguistics.

Pantel, Patrick and Dekang Lin. 2002. Document clustering with committees, In Proceedings of SIGIRO2, Tampere, Finland.