Probabilistic Parsing of Unrestricted English Text, With a Highly–Detailed Grammar

Ezra Black,
Stephen Eubank,
Hideki Kashioka
ATR Interpreting Telecommunications Laboratories
2-2 Hikaridai
Seika-cho, Soraku-gun
Kyoto, Japan 619-02
black@atr.itl.co.jp
kashioka@atr.itl.co.jp
eubank@atr.itl.co.jp

David Magerman
Renaissance Technologies Corp.
25 East Loop Road, Suite 211
Stony Brook, NY 11776 USA
magerman@rentec.com

19 June 1997

Summary

A grammar–based probabilistic parser is described, and experimental results are presented for the parser as trained and tested on a 676,000–word, highly varied treebank of unrestricted English text. Probabilistic decision trees are utilized as a means of prediction, and a grammar with about 3000 semantic–and–syntactic tags, and 1100 non–terminal node labels supplies detailed linguistic information. Further such data is supplied for prediction purposes by thousands of questions about "raw" words, expressions, and the sentence as a whole. The rich information base used for parse prediction allows the system to parse in a domain–general, totally–open-vocabulary setting, and to output highly–detailed semantic as well as syntactic information for sentences processed. Finally, a statistical procedure is described for converting less–detailed into more–detailed treebank, for use in increasing parser accuracy via much larger training treebanks.

Subject Areas: statistical parsing; automatic treebank conversion; semantic and syntactic analysis of text

1. INTRODUCTION

This article describes a grammar–based probabilistic parser, and presents experimental results for the parser as trained and tested on a large, highly varied treebank of unrestricted English text. Probabilistic decision trees are utilized as a means of prediction, roughly as in (Jelinek et al., 1994; Magerman, 1995), and as in these references, training is supervised, and in particular is treebank–based. In all other respects, our work departs from previous research on broad–coverage probabilistic parsing, which either attempts to learn to predict grammatical structure of test data directly from a training treebank (Brill, 1993; Collins, 1996; Eisner, 1996; Jelinek et al., 1994; Magerman, 1995; Sekine and Grishman, 1995; Sharman et al., 1990), or employs a grammar and sometimes a dictionary to capture linguistic expertise directly (Black et al., 1993a; Grinberg et al., 1995; Schabes, 1992), but arguably at a less detailed and informative level than in the research reported here.

In what follows, Section 2 explains the contribution to the prediction process of the grammar and of the lexical generalizations created by our grammarian. Section 3 shows, from a formal standpoint, how prediction is carried out, and more generally how the parser operates. Section 4 presents experimental results. Finally, Section 5 details our efforts to radically expand the size of our training corpus by employing techniques of treebank conversion.

2. HOW THE GRAMMAR AND LEXICAL GENERALIZATIONS HELP

2.1. How the Grammar Helps

Figure 1 shows a sampling of parsed sentences from the one-million-word ATR/Lancaster Treebank of General English (Black et al., 1996), which we employ for training, smoothing and testing our parser. The Treebank consists of a correct parse for each sentence it contains, with respect to the ATR English Grammar.[1] Every non-terminal node is labelled with the name of the ATR English Grammar rule[2] that generates the node; and each word is labelled with one of the 2843 tags in the Grammar's tagset.[3] Together, the bracket locations, rule names, and lexical tags of a Treebank parse specify a unique parse within the Grammar. In the Grammar parse, rule names and lexical tags are replaced by bundles of feature/value pairs. Each node contains values for 66 features, and there are 12 values per feature, on average.

Prediction in our parser is conditioned partially on questions about feature values of words and non-terminal nodes. For instance, when we predict whether a constituent has ended, we ask how many words until the next finite verb; the next comma; the next noun; etc. In tagging, we ask if the same word has already occurred in the sentence, and if so, what its value is for various features.

By labelling Treebank nodes with Grammar rule names, and not with phrasal and clausal names, as in other (non-grammar-based) treebanks (Eyes and Leech, 1993; Garside and McEnery, 1993; Marcus et al., 1993), we gain access to all information provided by the Grammar regarding each Treebank node.

It would be difficult to attempt to induce this information from the Treebank alone. The parent of a rule in the Grammar often contains feature values that are not derived from any of its children. Further, the parent inherits some feature values from one child, and some from another. Each rule in the Grammar is associated with a primary and secondary head, and head information is passed up the parse tree. Finally, extensive Boolean conditions are imposed on the application of each individual rule. These conditions are intended to permit only useful applications of a given rule, and reflect experience gained by parsing millions of words with the Grammar, and crucially, by generalizing this experience in ways believed appropriate.

Since the ATR English Grammar was created specifically for use in machine parsing, some of its features are designed expressly to facilitate parse prediction. For example, the feature

[1]On the ATR English Grammar, see below; for a detailed description of a precursor to the Grammar, see (Black et al., 1993a).

[2]There are 1155 rules in the Grammar.

[3]See (Black et al., 1996).

```
[start [sprpd1 [sprime4 [sd1 [nbar6 It_PPH1 nbar6]
[vbar2 [o8 has_VHZ o8] [v2 meant_VVNMEAN [nbar12 [j1 great_JJDEGREE j1]
[n1a savings_NN2MONEY n1a] nbar12] v2] vbar2] sd1]
[iebar2 ,_, [i1e [pr1 [rmod1 [r2 both_RRCONCESSIVE r2] rmod1]
[p1 in_IIIN [coord1 [nbar1 [n1a time_NN1TIME n1a] nbar1]
[coord3 [cc3 [cc1 &_CCAMP cc1] cc3]
[nbar1 [n1a gas_NN1SUBSTANCE n1a] nbar1] coord3] coord1] p1] pr1] i1e]
iebar2] sprime4] [rand3 !_! "_"R rand3] sprpd1] start]

[start [quo (_( [sprpd23 [sprime2 [ibbar2 [r2 Please_RRCONCESSIVE r2] ibbar2]
[sc3 [v4 Mention_VVIVERBAL-ACT [nbar4 [d1 this_DD1 d1]
[n1a coupon_NN1DOCUMENT n1a] nbar4] [fa1 when_CSWHEN
[v1 ordering_VVGINTER-ACT v1] fa1] v4] sc3] sprime2] sprpd23] )_) quo] start]

[start [sprpd22 [coord3 [cc3 [cc1 OR_CCOR cc1] cc3]
[nbar13 [d3 ONE_MC1WORD d3] [j1 FREE_JJSTATUS j1] [n4 [n1a FANTAIL_NN1ANIMAL n1a]
[n1a SHRIMPS_NN1FOOD n1a] n4] nbar13] coord3] sprpd22] start]
```

Figure 1: Three ATR/Lancaster English Treebank Sentences: One from Credit Union Brochure, and Two (Non-Sequential) from Chinese Take-Out Food Flier

"np_modification" helps to predict attachment events by carrying up to the top node of each noun phrase, data as to how much more modification the noun phrase can probably take. At one extreme, a noun phrase may not have been modified at all so far, and so, other things being equal, it is a prime target for post-modification. At the other extreme, it may already have been modified in a way that tends not to permit further modification, such as a noun phrase followed immediately by a postmodifying comparative phrase ("Such as can understand the topic (may attend)"; "More reasons than you can imagine (were adduced)").

Another feature of this type is "det_pos", which reveals, concerning a noun phrase, whether it includes a determiner phrase, and if so, what type. Determinerless noun phrases tend to have different chances of occurring in certain grammatical constructions than noun phrases with determiners, and this feature makes it possible for our models to take account of this tendency. Note that it is far from trivial to capture and then percolate this information up a treebank parse without a grammar: demarcation of the determiner phrase in each case is involved, along with identification of the type of determiner phrase, and other steps.

The ATR English Grammar is particularly detailed and comprehensive, and this both helps in parse prediction and enhances the value of output that is correctly parsed by our system. For instance, complete syntactic and semantic analysis is performed on all nominal compounds, e.g. "the Third Annual Long Branch, New Jersey Rod and Gun Club Picnic and Turkey Shoot", or "high fidelity equipment". Further, the full range of attachment sites is available within the Grammar for sentential and phrasal modifers, so that differences in meaning can be accurately reflected in parses. For instance, in "She didn't attend because she was tired, and didn't call for the same reason," the phrases "because she was tired" and "for the same reason" should probably postmodify their entire respective verb phrases, "didn't attend" and "didn't call", for maximum clarity. A full range of attachment sites are available in the Grammar, are used precisely in the Treebank, and are required to be handled correctly by our parser for its output to be considered correct.

2.2. How Lexical Generalizations Help

Prediction in our parser is conditioned not only on questions about feature values of words and non-terminal nodes, but also on questions about "raw" words, wordstrings, and whole sentences.

One category of contextual question asks about characteristics of a sentence as a whole. For instance, very short "sentences" in our training data tend to be free-standing noun phrases or other non-sentential units. Many of these are titles, speaker-turn indicators, etc. So we ask about the length of the overall "sentence" in all models. In tagging, for instance, there tend not to be any finite verbs in these contexts, and this fact helps with the task of differentiating, say, preterit forms from past participles functioning adjectivally, e.g. "Said plaintiff and plaintiff's counsel:". Similarly, the first and last words of a sentence can be powerful predictors. If the first word of a sentence is a typical beginning for sentential premodifying phrases (e.g. "Since"), and if there is just one comma in the sentence, and that comma occurs in the first quadrant, then there is a good chance that the overall structure of the sentence is: premodifying phrase, then main clause.

Effective questions about words and expressions, for the purpose of predicting the semantic portion of the lexical tags, are essential to the success of our models. One strategy we utilize is to identify contexts strongly associated with a given semantic event. For instance, the context: FirstName "X" LastName (e.g. Edward "Stubby" Smith) is one of many that are associated with the semantic category NickName.

2.3. Formulating Grammar and Lexical Questions For Prediction

We have developed a flexible language for formulating grammar-based and lexically-based questions about Treebank text. The answers to these questions are made available to the models in our parser.

The language provides facilities for navigating a parse tree, determining feature values of a given node, and making simple boolean or arithmetic computations. In addition, it allows us to translate answers returned by the question into a more natural format for input to the decision-tree models.

The language provides easy access to word and tag nodes at any offset from the beginning or end of the sentence. It also provides a reference position—the "current" node, i.e. the node about which a prediction is being made. It is easy to navigate from any node to previous nodes, parent/child nodes, and word/tag nodes relative to the node's constituent boundaries. The navigational commands are recursive, so that, for example, one can arrive at a grandchild of a node by asking about a child's child.

There is nothing in the language itself which restricts the context which can be used in models. For example, changing a bigram tagger into a trigram tagger requires only adding questions about the additional nodes. More generally, the ability to ask questions about the entire sentence (and, in the future, document), means that the "context" is of variable length.

Every question has access to the current parse state, which contains everything known or predicted about the parse tree up to the time the question is asked. Any of this information is available for a selected node. For word nodes, this includes membership on vocabulary lists, whether the word contains various prefixes, suffixes, substrings, etc. In addition, for tag and nonterminal nodes, the name of the label and the values of all the Grammar's features (including those based on information propagated up the parse tree from lower down) at that node are also available. Finally, for nonterminal nodes, general information about the number of children, span, constituent boundaries, etc. is available.

Answers to the questions are of various types: Boolean, categorical, integer, sets of integers. But we transform all these types of answers into binary strings. Some transformations are obvious. Boolean values, for example, are mapped to a single bit. Other transformations are based on clustering, either expert or automatic. For example, the sets of tags and rule labels have been clustered by our team grammarian, while a vocabulary of about 60,000 words has been clustered by machine (Brown et al., 1992; Ushioda, 1996a; Ushioda, 1996b).

3. HOW PREDICTION IS CARRIED OUT

3.1. System Design

The ATR parser is a probabilistic parser which uses decision–tree models. A parse is built up from a succession of *parse states*, each of which represents a partial parse tree. Transition between states is accomplished by one of the following steps: (1) assigning syntax to a word; (2) assigning semantics to a word; (3) deciding whether the current parse tree node is the last node of a constituent; (4) assigning a (rule) label to an internal node of the parse tree. Note that the first two steps together determine the tag for a word, and the third determines the topology of the tree. Working from the bottom up, left to right, constrains the parser to produce a unique derivation for each parse state. Alternatively, we can tag the entire sentence first, then work from tags up, left to right, which also yields a unique derivation for each parse state.

Statistical models corresponding to each type of step provide estimates of the probability of each step's outcome.[4] Each model uses as input the answers to a set of questions about context designed specifically for that model by our team grammarian, using the language described in Section 2.3. Thus the probability of each decision depends on features extracted from the context, including information about any word(s) in the sentence and any tags and parse structure already predicted. The estimated probability of any parse state is the product of the probabilities of each step taken to reach that state. Strictly speaking, we estimate relative *likelihoods* rather than probabilities, since we make no attempt to normalize over all possible parses for a given sentence.

Given a set of models for estimating the probabilities of parse steps, the problem of predicting a parse reduces to searching the space of possible parses for the most likely one. We use a chart parser (Kasami, 1965) to build a compact representation of all legal parses for the sentence, which in turn constrains the search to consider only those parse steps guaranteed to lead to a complete (legal) parse. Even so, because the Grammar generates a large number of parses for each sentence,[5] it is not feasible to rank the parses exhaustively. Fortunately, incomplete parse states are assigned probabilities, which can be used to guide a search by ruling out unlikely parses without constructing the complete parse. We have found that a greedy search, which chooses the most likely outcome for each parsing step, usually finds a good candidate parse. Occasionally, though, choosing a less likely step at one point leads to a parse with higher overall likelihood. To allow for this possibility, we use the greedy candidate parse to "seed" the stack-based decoder described in (Jelinek, 1969).

There is some freedom in the order in which the parsing steps are taken. The context in which a model makes its prediction includes any parts of the parse tree which have already been built. Hence, the order chosen determines what information is available to each model. We choose to tag the entire sentence first, producing an N-best list of tag sequences. Specifically, starting from a sequence of words, we first tag the sentence as follows:

- estimate the probability for each part-of-speech of the first word;

[4]For efficiency we break down the semantic model further into a set of models, one for each syntactic category.
[5]Its Parse Base (Black et al., 1993a) is 1.76.

- choose one or more most likely parts-of-speech;
- estimate the probability for each tag for the first word, given the part-of-speech decision(s) made above;
- choose one (or several) likely tag(s);
- repeat the steps above for each word in the sentence.

Next, starting from the tag of the first word, which is the left-most leaf node of the parse tree, we take the following steps:

- estimate the probability that the current node of the parse tree is the last child of its parent (e.g. the probability that a constituent ends at this node);
- if a constituent is deemed to end at this node, estimate the probability of possible rule labels for that consitutent, i.e. of only those rules which are known to lead to legal parses; make that node the current node; and return to the first step;
- otherwise, make the top of the next subtree to the right the current node and return to the first step.

This approach decouples the search over tag sequences from the search over parse trees.

3.2. Decision–Tree Models

The parser requires models which estimate the probability of membership in a class given an input vector. We use class probability trees, a slight modification of classification trees, as described in (Breiman et al., 1984; Quinlan, 1986; Bahl et al., 1983), with a few enhancements. We can choose among several different standard splitting criteria for the trees. The trees are pruned using the minimal cost–complexity algorithm (Breiman et al., 1984). In addition, estimates for probability distributions are smoothed using the Forward-Backward algorithm (Baum, 1972).

The models are trained using bitstring answers to questions about each state encountered while parsing each sentence in the training set. We build binary trees, in which each node can split the data based on the value of any bit in the bitstring. There are situations in which an entire question does not apply—for example, a question about the previous word when the first word of a sentence is under consideration. These situations are flagged so that the decision tree will split out this data before it asks about any of the bits in the answer to this question.

4. EXPERIMENTAL RESULTS

4.1. Evaluation Methodology

In our view, any effective evaluation methodology for automatic grammatical analysis must confront head–on the problem of multiple correct answers in tagging and parsing. That is, it is often the case that there is more than one "correct tag" for a word in context, where that word could be considered to be functioning as: a proper or a common noun; an adjective or a noun; a participle or an adjective; a gerundial noun or a noun;[6] an adverbial particle or a locative adverb; and even an adjective or an adverb. This is true even where there are highly detailed and well–understood

[6]terminology of (Long, 1961), for e.g. a *sleeping* pill vs. to make a good *living*

guidelines for the application of each tag to text. And obviously the existence of multiple correct taggings for a word is to be expected a fortiori where a highly ramified system of semantic categories is involved. It follows that multiple correct parses exist for many sentences, since by definition any change in tag means a change in parse. But other sources of multiple correct parses exist as well, and range from, say, several equally good attachment sites within a parse for a given modifier, even given full document context, to cases where the grammar itself provides several equally good parses for a sentence, through the presence of normally independent rules whose function nonetheless overlaps to some degree.

Barring the recording of the set of correct tags for each word, and of the set of correct parses for each sentence, in a treebank, the next-best solution to the problem of multiple correct answers is to at least provide such a recording in one's test set, i.e. to provide a "gold standard" test set with all correct tags and parses for each word in context. This is the solution that was adopted in creating the ATR/Lancaster English Treebank.

The way we evaluate our tagger is to compare its performance to the set of correct tags for each word of each sentence of our "gold standard" test data. Thus, in all cases we are able to take into account the full set of "correct" answers.[7] Since 32% of running words in our test data have 2 or more correct tags, potential differences in performance evaluation are large vis-a-vis traditional metrics.[8]

Similarly, in the case of the parser, we evaluate performance against a special "gold standard" test set which lists every correct parse with respect to the Grammar for each test sentence. We utilize two measures. First is exact match with any correct parse listed for the sentence. Second is "exact syntactic match": exact match with the bracket locations and rule names only. Notice that in a parse considered correct by our second metric, the syntax[9] of all tags must be correct.

The average number of different correct "exact syntactic matches"[10] per sentence in our test data is 3. Among test-data sentences, 72% have more than one correct exact syntactic matches, and 32% have 5.[11] For critiques of other approaches to broad-coverage parser and tagger evaluation, see (Black, 1994).

It is worth inquiring how well expert humans do at the parsing task that we are attempting here by machine. Accordingly, we present statistics below on the consistency and accuracy of expert humans at parsing using the ATR English Grammar. The ATR/Lancaster treebanking effort features a grammarian, who originated the Grammar, and a treebanking team, who apply the Grammar to treebank text. We can therefore distinguish two different types of evaluation as to how well expert humans do at parsing using the Grammar: consistency and accuracy. Consistency is the degree to which all team members posit the identical parse for the identical sentence in the identical document of test data. Accuracy is the expected rate of agreemnt between a treebanker and the grammarian on parsing a given sentence in a given document of test data.

In a first experiment to determine consistency, we asked each of the three team members to declare either correct or incorrect a particular parse for a sentence of test data. The parses had

[7]We limit the set of correct tags to five tags; however, for only 2% of running words of test data were as many as 5 tags provided by our human experts; so in general, we are accounting for "all correct tags" for the given word in context.

[8]Actually, so far, we have found about a 10% improvement both in tagging and parsing results when we test against the full set of correct answers, as opposed to testing against the single answer in the original treebank parse of a sentence.

[9]and often some of the semantics

[10]i.e. parses with a unique set of bracket locations and bracket labels (Grammar rule names)

[11]Five is the maximum number of correct exact syntactic matches that we ask our treebankers to supply per sentence, for test data.

length	# sentences	top	top 20	cross	constits/sent
1-10	1044	81.8%	95.0%	89.1%	7.6
11-15	248	30.2%	72.6%	43.1%	23.9
16-23	201	17.4%	48.3%	28.4%	34.2

Table 1: Parsing from text which starts out correctly tagged: percentage of parses which exactly match one of the human-produced parses. "Cross" indicates percentage of test-data sentences whose top-ranked parse contains 0 instances of "crossing brackets" with respect to the most probable treebank parse of the sentence.

been generated with respect to our Grammar, by trained humans, but whose skills at parsing with the Grammar were not as good as those of our three team members. 384 sentences of test data were utilized. The result was a 6.7% expected rate of disagreement among the team members on this task.[12] In a second consistency experiment, we located all sentences occurring twice or more in the Treebank; if there were more than two duplicates, we selected just two at random. We then determined the number of duplicate-sentence pairs that were exact matches in terms of the way they were parsed and tagged. 76% of these 248 sentence pairs were such exact matches.[13]

Finally, in an experiment to determine accuracy of our team members' parsing using the Grammar, the ATR grammarian scored for parsing and tagging accuracy some 308 sentences of Treebank data from randomly-selected Treebank documents.[14] The result of this scoring was a 8.4% expected parsing error rate.[15]

4.2. Experimental Results

As discussed in 3.1, our first step in parsing is to tag each sentence. The tagger currently produces an exact match 74% of the time for the 47,800-word test set, comparing against a single tag sequence for each sentence.[16] We present parsing results both for text which starts out correctly tagged (Table 1)[17] and for raw text (Table 2). Results for parsing from raw text are given for both the exact-match and exact-syntactic-match criteria described in 4.1.

The performance of the parser on short sentences of correctly tagged data is extrememly good. We feel this indicates that the models are performing well in scoring the parses.

The results deteriorate rapidly for longer sentences, but we believe the problem lies in the search procedure rather than the models. A measure of the performance of a search is whether it

[12]In a parallel experiment to determine consistency on tagging, we asked each of the three team members to choose the first correct tag from a ranked list of tags for each word of each sentence of test data. These ranked lists were hand-constructed, and an effort was made to make them as difficult as possible to choose from. About 4,800 words (152 sentences) of test data were utilized. The result was a 3.1% expected rate of disagreement among the team members on the exact choice of tag.

[13]Of these 248 sentence pairs, 85% were exact matches in terms of the way they were tagged.

[14]Actually, the documents were selected from our "main General-English Treebank" of 800,000 words.

[15]i.e.. the parse was wrong if even one tag was wrong; or, of course, if a rule choice was wrong. For the tags assigned to the roughly 5000 words in these 308 sentences, expected error rate was 2.9%. Essentially none of these tagging errors had to do with the use of the syntactic portion of our tags; all of the errors were semantic; the same was true in the two tagging consistency experiments related above.

[16]As noted in 4.1 fn. 8, our experience indicates that we can expect a roughly 10% improvement in this score when we compare performance against "golden-standard" test data in which all correct answers are indicated; this would bring our tagging accuracy into the 80-percent area.

[17]For the definition of the term "crossing brackets" used in Table 1, see (Harrison et al., 1991).

Length	exact match		syntactic exact match	
	top	top 10	top	top 10
1-10	34.5%	40.1%	50.4%	62.3%
11-15	1.2%	3.6%	11.3%	25.6%

Table 2: Parsing from raw text: percentage of parses which exactly match one of the human-produced parses ("exact match") or which match bracket locations, rule names, and syntactic part-of-speech tags only ("syntactic exact match").

Feature	IBM Manuals Treebank	ATR/Lancaster Treebank
Vocabulary Type	Restricted	Open
Vocabulary Size (Training Corpus)	3,000	35,952
Domain	IBM Computer Manuals	Unrestricted English
Tagset Size	193	2,843 (440 Syntax-Only)
Nonterminal Labels	17	1,155
Test-Data Source	?	Entire Documents
Training Set Size (in words)	about 438,000	676,401
Test Set Size (in words)	about 25,000	47,800
Average Sentence Length (Training Corpus)	about 15	15.8
Average Sentence Length (Test Corpus)	16.9	13.1
Number of Constits in 20-Word Sentence	about 11	about 34

Table 3: Comparison of IBM Manuals and ATR/Lancaster General-English Treebanks

suggests any candidates which are as likely as the correct answer. If not, the parser has erred by "ommission" rather than by "commission": it has ommitted the correct parse from consideration, but not because it seemed unlikely. It is entirely possible that the correct parse is in fact among the highest-scoring parses. These types of search error are non-existent for exhaustive search, but become important for sentences between 11 and 15 words in length, and dominate the results for longer sentences.

The results in Table 2 reflect tagging accuracy as well as the performance of the parser models per se. Note that tagging accuracy is quoted on a per-word basis, as is customary. From previous work, we estimate the accuracy of the tagger on the syntactic portion of tags to be about 94%. Thus there is typically at least one error in semantic assignment in each sentence, and an error in syntactic assignment in one of every two sentences. It is not surprising, then, that the per-sentence parsing accuracy suffers when parses are predicted from raw text.

Clearly the present research task is quite considerably harder than the parsing and tagging tasks undertaken in (Jelinek et al., 1994; Magerman, 1995; Black et al., 1993b), which would seem to be the closest work to ours, and any comparison between this work and ours must be approached with extreme caution. Table 3 shows the differences between the treebanks utilized in (Jelinek et al., 1994) on the one hand, and in the work reported here, on the other.[18] Table 4 shows relevant

[18]Figures for Average Sentence Length (Training Corpus) and Training Set Size, for the IBM Manuals Corpus, are approximate, and come from (Black et al., 1993a).

Length	# sentences	top	top 20	cross	constits/sent
1-10	447	55.9%	80.8%	91.5%	4.6
11-15	436	47.5%	76.6%	80.7%	8.2
16-23	430	21.6%	48.8%	56.5%	11.3

Table 4: Parsing results reported by Jelinek et. al. for IBM Manuals task; see Table 3 above

parsing results by (Jelinek et al., 1994). Even starker contrasts obtain between the present results and those of e.g. (Magerman, 1995; Black et al., 1993b), who do not employ an exact-match evaluation criterion, further obscuring possible performance comparisons. Obviously, no direct comparisons of the results of Tables 1–2 with previous parsing work is possible, as we are the first to parse using the Treebank.

In our current research, we are emphasizing the creation of decision-tree questions for predicting semantic categories in tagging, as well as continuing to develop questions for syntactic tag prediction, and for our rule-name-prediction model.

5. TOWARDS RADICALLY EXPANDING TRAINING-SET SIZE VIA TREEBANK CONVERSION

5.1. Introduction

As an additional means of improving the accuracy of our parser, we have been working towards effecting a dramatic increase in the size of our training treebank, via treebank conversion techniques. We employ a statistical method for converting treebank from a less-detailed format—and we have chosen the IBM/Lancaster Treebank (Eyes and Leech, 1993; Garside and McEnery, 1993) as a first representative of such treebanks—to a more-detailed format, that of the ATR/Lancaster Treebank.

There has been very little previous work on treebank conversion. (Hughes et al., 1995) describe an effort to hand-annotate text using the tagging schemes employed in various different treebanks, as a preliminary to attempting to learn, in a way to be determined, how to convert a corpus automatically from one style of tagging markup to another. (Wang et al., 1994) take on the problem of converting treebank conforming to their English grammar into a format conforming to a later version of the same grammar, and report a conversion accuracy of some 96% on a 141,000-word test set. They employ a heuristic which scores source-treebank/target-treebank parse pairs based essentially on the percentage of identically-placed brackets in the two parses. However, their target grammar[19] generates only 17 parses on average per sentence of test data. Although they exhibit no parses with respect to their grammars, it can be assumed that they feature only rudimentary tag and non-terminal vocabularies.

The problem we face in learning to convert IBM/Lancaster Treebank parses into ATR/Lancaster Treebank parses is rather more difficult than this. For instance, as noted in 3.1, the Parse Base of the ATR English Grammar, which generates the parses of the ATR/Lancaster Treebank, is 1.76, which means that on average, the Grammar generates about 200 parses for 10-word sentence; 2000 parses for a 15-word sentence, and 70,000 parses for a 20-word sentence. Further, far from featuring a rudimentary set of lexical tags and non-terminal node labels, the ATR/Lancaster Treebank utilizes

[19]and presumably their source grammar as well

```
However_RR ,_, [N GM_NNJ N] [V has_VHZ announced_VVN [N plans_NN2
[Ti to_TO cut_VVO back_RP on_II [N frames_NN2 N] [P in_II
[N efforts_NN2 [Ti to_TO [V [V& conserve_VVO [N space_NN1 N] V&]
and_CC [V+ reduce_VVO [N weight_NN1 N]
[P in_II [N new_JJ cars_NN2 N] P] V+] V] Ti] N] P] Ti] N] V] ._.
```

```
[start [sprpd1 [sprime2 [ibbar1 [i1g [r2 However_RRCONCESSIVE r2] i1g]
,_, ibbar1] [sd1 [nbar1 [n1a GM_NP1FRMNM n1a] nbar1] [vbar2 [o8
has_VHZ o8] [v4 announced_VVNVERBAL-ACT [nbarq4 [nbar1 [n1a
plans_NN2PROGRAM n1a] nbar1] [i1b [t1 [vibar1 to_TO [v36 cut_VVIALTER
[r2 back_RP r2] [p1 on_IION [nbar1 [n1a frames_NN2DEVICE-PT n1a]
nbar1] p1] v36] vibar1] t1] i1b] nbarq4] [p1 in_IIIN [nbarq4 [nbar1
[n1a efforts_NN2INTER-ACT n1a] nbar1] [i1b [t1 [vibar1 to_TO [v2 [v41
[v40 conserve_VVIHELP [nbar1 [n1a space_NN1MEASURE n1a] nbar1] v40]
and_CCAND [v40 reduce_VVIALTER [nbar1 [n1a weight_NN1MEASURE n1a]
nbar1] v40] v41] [p1 in_IIIN [nbar12 [j1 new_JJTIME j1] [n1a
cars_NN2DEVICE n1a] nbar12] p1] v2] vibar1] t1] i1b] nbarq4] p1] v4]
vbar2] sd1] sprime2] ._. sprpd1] start]
```

Figure 2: IBM/Lancaster Treebank and ATR/Lancaster Parses For Same Sentence

roughly 3,000 lexical tags and about 1,100 different non-terminal node labels,[20] as mentioned in 2.1. Figure 2 shows a parse for a sample sentence, first from the IBM/Lancaster Treebank, and next from the ATR/Lancaster Treebank. An impression of the difficulty of the treebank conversion task undertaken here can be gained by closely contrasting the two parses of this Figure.

143,837 words included in the IBM/Lancaster Treebank—35,575 words of Associated Press newswire and 108,262 words of Canadian Hansard legislative proceedings—were treebanked with respect to the ATR English Grammar, in the exact same manner as the data in the ATR/Lancaster Treebank. We will refer to the IBM/Lancaster Treebank version of this data as the parallel corpus. As a preliminary step to treebank conversion, we aligned the parallel and ATR corpora. 87.3% of the parallel data—125,530 words—aligned essentially perfectly, and for the work reported here, we decided to operate only on this satisfactorily-aligned data.

5.2. The Treebank Conversion Problem

Ideally, our treebank-conversion models should take full advantage of data in the full target treebank (i.e. the full ATR/Lancaster Treebank) as well as the parallel corpus. A direct model of the conditional probability of the ATR parse given the source-treebank parse, $p(A|F)$, uses only data in the parallel corpus. A more efficient use of data would be to build two models: one to estimate the likelihood of an ATR parse, $p(A)$, given raw text; the other to estimate $p(F|A)$. Then,

[20]actually, rules names with respect to the ATR English Grammar; cf. 2.1

using Bayes' rule, one would write $p(A|F)$ as:

$$p(A|F) \propto p(F|A)p(A) \tag{1}$$

The model for $p(F|A)$ uses only the parallel corpus, but the model for $p(A)$ makes full use of the data in the ATR treebank.

In our software environment, this approach would require constructing a feature–based grammar for the source treebank. A simpler, but probably adequate approach would combine the two models $p(A)$ and $p(A|F)$ heuristically, using $p(A|F)$ to rescore the N best parses found by the model $p(A)$. The top-ranked candidate from the rescored parses is selected as the ATR parse. This way takes advantage of both data sets, though not as efficiently as the Bayesian approach. We have chosen to explore the problem using an even simpler approach: ignoring the ATR treebank and working only within the model for $p(A|F)$. This yields lower bounds on potential accuracy at low cost.

We also considered filtering the parses considered by the ATR parser to ensure they satisfied certain constraints implied by the source–treebank parse. This proved to be impractical because the constraints were not "hard", i.e. the exact circumstances in which they should be applied were difficult to determine. Instead, we relied on the models to learn the constraints and the conditions for their application directly from the data. However, the issue of applying such constraints is specific to the two treebanks being used; there may well be cases in which such constraints are not hard to develop.

The source–treebank–to–ATR conversion model was built using the same system described in Sections 2 and 3, the sole difference being that the question language was extended to allow for questions about the source treebank. Since the topology of the parallel tree may be very different from that of the ATR parse tree, it is not obvious what the analog of a node in the ATR tree is. We chose to use the "least enclosing" node: that is, the lowest (non-preterminal) node in the parallel tree which spans (at least) the set of words spanned by the node in the ATR parse.

5.3. Decision–Tree Questions Asked

We ask all decision–tree questions in our treebank–conversion models that we do normally in parsing with the ATR English Grammar.[21] We then add further questions which ask about the source–treebank parse for the sentence being processed.

We use an extremely basic set of question–language functions in querying the structure of the source–treebank parse. These permit us to ask about the least–enclosing node, and about children and parents of this source–treebank–parse node, or of its children or parents, to any level of structure. What we can ask about a node in the source–treebank parse is either what its non-terminal label is, or how many children it has. In addition, we are able to ask whether there is a constituent in the source–treebank parse with the identical span as a given node of an ATR parse; and if so, what its non–terminal label is, or how many children it has. Similarly, we can ask about constituents that "cross" a given node of an ATR parse. Finally, we can ask about the tag of any word in the source–treebank parse.

There is much farther that we can go in exploiting the information in the source–treebank parse to aid in predicting the ATR parse. For instance, we can define and query grammatical relations such as clausal subject and main verb. We can even define and query notions like "headword" with respect to the source–treebank parse, although this would involve appreciable work. Furthermore, carrying over to the source–treebank environment question types that seem helpful when asked about ATR parses will not be difficult.

[21] Cf. Section 2

Length	treebank conversion		parser	
	top	top 10	top	top 10
1-10	61.5	96.2	63.0	92.6
11-15	46.7	66.7	33.3	73.3

Table 5: Parsing from text which starts out correctly tagged: percentage of parses which exactly match the single parse in the treebank, for a 6,556–word test set. "Treebank–conversion" models are trained on 118,489 running words of ATR/Lancaster Treebank, together with aligned IBM/Lancaster Treebank. "Parser" models are trained on 676,401 running words of ATR/Lancaster Treebank alone.

5.4. Experimental Results

Evaluation Methodology We evaluate treebank conversion to ATR–Treebank format in the same way as we evaluate the parser when it is trained in the normal manner (cf. 4.1), except that test data consists of ATR–Treebank–format documents of which we also possess aligned source treebank (in this case, IBM/Lancaster–Treebank) versions. In the performance results cited below, however, we show exact match only with the single correct parse of the test treebank, rather than with any one of the correct parses indicated in the "golden standard" version of the test set.

Experimental Results Table 5 displays exact–match parsing results for a normal 6,556–word test set[22]. Crucially, the amount of training data here, 118,489 words, is only 17.5% as large as for the models of Tables 1–2. Considering the simplicity of the approach, we think these results constitute a proof of principle for the idea of treebank conversion. They indicate that we can build treebank conversion models of accuracy comparable to the current parser using much less data. Of course, the results here do not include models used in tagging. The treebank conversion models tag with an accuracy of 62.8%. A detailed examination of those models shows that the syntactic models are better than the parser's, while the semantic models are worse. This is to be expected, because the IBM/Lancaster Treebank contains a great deal of relevant information about the syntax, but not so much about the semantics of the sentences they contain. One idea, therefore, is to utilize large–scale treebank conversion in the tagging domain to overcome the problem noted in 4.2, that even with 94% accuracy at strictly syntactic tagging (i.e. effectively, on tagging with our 440–tag syntax–only tag subset), approximately one word is syntactically mistagged every two sentences, leading to an increased error rate at exact–syntactic–match parsing. A second direction which suggests itself is to pursue our scaled–down approach to treebank conversion, but with more training data than we have used so far. Third, we may decide to implement the more laborious two–model approach desribed in 5.2.[23] Overall, we expect that conversion models which take full advantage of the existing database as well as of the parallel corpus as outlined above should produce data of high enough quality to use as training data for our parser.

[22]i.e. not for a "golden standard" test set as described in 4.1, in which all parses are indicated for each test sentence

[23]It seems worth mentioning that future large–scale treebank–creation efforts would probably benefit from constructing parallel data with respect to other large treebanks, right from the start.

6. ACKNOWLEDGEMENTS

We wish to thank Joshua Goodman and John Lafferty for their contributions to the treebank conversion work reported here; Akira Ushioda for his implementation of the Brown word–clustering algorithm; and Craig MacDonald and Toyomi Saiga for their contributions to our work overall.

REFERENCES

L. Bahl, P. Brown, P. deSouza, and R. Mercer. 1989. A tree–based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36.7:1001–1008.

L. Baum. 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of markov processes. *Inequalities*, 3:1–8.

E. Black, S. Eubank, H. Kashioka, R. Garside, G. Leech, and D. Magerman. 1996. Beyond skeleton parsing: producing a comprehensive large–scale general–English treebank with full grammatical analysis. In *Proceedings of the 16th Annual Conference on Computational Languistics*, pages 107–112, Copenhagen.

E. Black. 1994. A new approach to evaluating broad–coverage parser/grammars of English. In *Proceedings of the International Conference on New Methods in Language Processing.* Manchester, UK. September, 1994.

E. Black, R. Garside, and G. Leech, Editors. 1993. *Statistically–Driven Computer Grammars Of English: The IBM/Lancaster Approach.* Rodopi Editions. Amsterdam.

E. Black, F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, S. Roukos. 1993. Towards History-Based Grammars: Using Richer Models For Probabilistic Parsing. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics.* Columbus, Ohio. Also in *Proceedings of the DARPA Speech and Natural Language Workshop*, 1992.

E. Brill. 1993. Automatic grammar induction and parsing free text: a transformation–based approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Languistics*, pages 259–265, Columbus, Ohio.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. 1984. *Classification and Regression Trees.* Wadsworth & Brooks/Cole, Monterey, CA.

P. Brown, V. Della Pietra, P. de Souza, J. Lai, R. Mercer. 1992. Class–based n–gram models of natural language. *Computational Linguistics*, 18.4:467–479.

M. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Languistics*, Santa Cruz.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th Annual Conference on Computational Languistics*, pages 340–345, Copenhagen.

E. Eyes and G. Leech. 1993. Syntactic Annotation: Linguistic Aspects of Grammatical Tagging and Skeleton Parsing. Chapter 3 of Black et. al. 1993.

R. Garside, G. Leech, G. Sampson, Editors. 1987. *The Computational Analysis of English.* London, Longman.

R. Garside and A. McEnery. 1993. Treebanking: The Compilation of a Corpus of Skeleton–Parsed Sentences. Chapter 2 of Black et. al. 1993.

D. Grinberg, J. Lafferty, and D. Sleator. 1995. A robust parsing algorithm for link grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*, Prague.

P. Harrison, S. Abney, E. Black, D. Flickenger, C. Gdaniec, R. Grishman, D. Hindle, R. Ingria, M. Marcus, B. Santorini, T. Strzalkowski. 1991. Evaluating Syntax Performance Of Parser/Grammars Of English. In *Proceedings of the Workshop On Evaluating Natural Language Processing Systems*, Association For Computational Linguistics.

J. Hughes, C. Souter, and E. Atwell. 1995. Automatic extraction of tagset mappings from parallel-annotated corpora. In *Proceedings of SIGDAT Workshop*, Dublin.

F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, S. Roukos. 1994. Decision tree parsing using a hidden derivation model. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 260–265, Plainsboro, New Jersey. Advanced Research Projects Agency.

F. Jelinek. 1969. A fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, 13:675–685.

T. Kasami. 1965. An efficient recognition and syntax algorithm for context–free languages. Scientific Report AFCRL–65–758. Air Force Cambridge Laboratory. Bedford, Massachusetts.

R. Long. 1961. *The Sentence and Its Parts.* University of Chicago Press. Chicago.

D. Magerman. 1995. Statistical decision–tree models for parsing. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, Massachusetts. Association for Computational Linguistics.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19.2:313-330.

J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1:80–106.

Y. Schabes. 1992. Stochastic lexicalized tree–adjoining grammars. In *Proceedings of the 15th International Conference on Computational Linguistics*, Nantes.

S. Sekine and R. Grishman. 1995. A corpus-based probabilistic grammar with only two non-terminals. In *Proceedings, International Workshop on Parsing Technologies*, 1995.

R. A. Sharman, F. Jelinek, and R. Mercer. 1990. Generating a grammar for statistical training. In *Proceedings, DARPA Speech and Natural Language Workshop*, Hidden Valley, Pennsylvania.

A. Ushioda. 1996. Hierarchical clustering of words. In *Proceedings of the 16th Annual Conference on Computational Languistics*, pages 1159–1162, Copenhagen.

A. Ushioda. 1996. Hierarchical clustering of words and application to NLP tasks. In *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 28–41, Copenhagen.

J. Wang, J. Chang, and K. Su. 1994. An automatic treebank conversion algortihm for corpus sharing. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 248–254, Las Cruces, New Mexico.