TR-IT-0309

The ATR General English Parser

Ezra Black

September 1999

Abstract

The work of the Statistical Parsing Group of Department 3 (Natural Lanfguage Processing), Interpreting Telecommunications Laboratory, ATR, over the period 1993–1999 is presented. **Keywords NLP natural language processing parsing English language tagging language modelling speech synthesis statistical methods machine learning**

Contents

1 Acknowledgements 1

2 Overview 3

3 Providing Training Data: Annotation System for English Text 4
- 3.1 Introduction . . . 4
- 3.2 General Description of the Treebank . . . 4
  - 3.2.1 Document Selection and Preprocessing . . . 4
  - 3.2.2 Scheme of Grammatical Annotation . . . 6
- 3.3 Producing the Treebank . . . 7
  - 3.3.1 The Software Backbone—GWBTool: A Treebanker's Workstation . . . 8
  - 3.3.2 Two–Stage Part–Of–Speech Tagging . . . 10
  - 3.3.3 The Annotation Process . . . 10
  - 3.3.4 Staff Training; Output Accuracy . . . 10
- 3.4 Conclusion . . . 11

4 Predicting Word Meaning and Function: Part–Of–Speech Tagging 11
- 4.1 Introduction: Building On The Successes To Date In Part–Of–Speech Tagging . . . 11
- 4.2 Real–World Part–Of–Speech Tagging . . . 13
  - 4.2.1 Tag Using Tagsets Of Increased Size And Complexity . . . 13
  - 4.2.2 Confront The Unknown–Word And –Tag Problems . . . 17
- 4.3 The Non–Dictionary . . . 19
- 4.4 The Model . . . 20
  - 4.4.1 Mututal Information (MI) Bits . . . 20
  - 4.4.2 Decision Trees . . . 20
  - 4.4.3 The Tagging Process . . . 21
  - 4.4.4 Example Questions . . . 22
- 4.5 Experimental Results . . . 22
- 4.6 Experiments In Tagging Improvement (1) . . . 24
- 4.7 Background . . . 26
- 4.8 The Experiments . . . 27
  - 4.8.1 Experimental Design . . . 27
- 4.9 Experimental Results . . . 29
  - 4.9.1 Window Size . . . 29
  - 4.9.2 Class-Specific Triggers . . . 30
- 4.10 Discussion . . . 30
- 4.11 Some Examples . . . 33
- 4.12 Conclusion . . . 33
- 4.13 Experiments In Tagging Improvement (2) . . . 34
- 4.14 Tagging Model . . . 35
  - 4.14.1 ME Model . . . 35
  - 4.14.2 Trigger selection . . . 36
- 4.15 The Constraints . . . 36
- 4.16 The Experiments . . . 38
  - 4.16.1 The Four Models . . . 38

4.16.2 Experimental Procedure 38
4.17 The Results 40
4.18 Conclusion 40

5 Predicting Meaning and Function of Phrases and Sentences: Parsing 41
5.1 Introduction 41
5.2 How the Grammar and Lexical Generalizations Help 41
5.2.1 How the Grammar Helps 41
5.2.2 How Lexical Generalizations Help 43
5.2.3 Formulating Grammar and Lexical Questions For Prediction 44
5.3 How Prediction Is Carried Out 44
5.3.1 System Design 44
5.3.2 Decision–Tree Models 46
5.4 Experimental Results 46
5.4.1 Evaluation Methodology 46
5.4.2 Experimental Results 48
5.5 Towards Radically Expanding Training–Set Size Via Treebank Conversion 50
5.5.1 Introduction 50
5.5.2 The Treebank Conversion Problem 52
5.5.3 Decision–Tree Questions Asked 52
5.5.4 Experimental Results 53

6 Application To AI Tasks: Upper–Bound Experimentation 54
6.1 Introduction 54
6.2 The Language Model (LM) 55
6.2.1 ME Model 55
6.2.2 Trigger selection 56
6.3 Linguistic Information 56
6.4 The Experiments 57
6.4.1 Experimental Procedure 57
6.4.2 Effect of Dataset Size 58
6.4.3 Effect of Adding Word Triggers 59
6.5 Discussion 59
6.6 Application To Speech Synthesis Tasks 60

1 Acknowledgements

What follows is a report on the research activities and outcomes, over the period 1993 through 1999, of the Statistical Parsing Group within Department 3 (Natural Language Processing) of the Interpreting Telecommunication Laboratory of ATR, Kyoto, Japan. While the nominal author of this report is E. Black, the work described is that of the entire Statistical Parsing Group, whose membership, during this six–year period, has included the following individuals:

- Applied mathematicians: Eubank–san (95–96); Finch–san (97–present)
- Computational linguists: Kashioka–san (93–present); Ushioda–san (95)
- Master programmers: MacDonald–san (96); Shalif–san (98–present)

- Consultants: Magerman–san (94-95); Lafferty–san (95–99)
- Grammarian: Black (93–present)
- Summer Students:
  - Saia–san (93–94) (Computer Science)
  - El Nahas–san (95) (Mathematics)
  - Goodman–san (95) (Computational Linguistics)
- Data Entry: Masui–san (95–97)
- Data Preparation (Treebanking):
  - Treebankers:
    * At ATR: Lemmon–Kishi–san (93–94); Murphy–san (93–94)
    * At Lancaster, UK: Bateman–san (94–97); Forrest–san (94–97); Willis–san (95-97)
  - Computing support at Lancaster: Garside–san; Hutchinson–san
  - Management support at Lancaster: Leech–san; Preston–san; Needham–san

Also integral to the work of the Statistical Parsing Group has been the cooperative research it has pursued with other ITL researchers, who are as follows:

- Speech Recognition, Department 1: Sagisaka–san; Zhang–san
- Speech Synthesis, Department 2: Campbell–san; Hebert–san; Srinivas–san

Since 1995, a research group has been in existence, under Kashioka–san, with the aim of transferring to Japanese the approach to English parsing to be described below. Nothing further will be said in this report of this work in Japanese, since this constitutes a separate research effort with different, though highly related, aims to those of the Statistical Parsing Group. The personnel of this research group has been as follows:

- Computational Linguist, Supervisor: Kashioka–san
- Computational Linguists, Grammarians: Kawata–san; Kinjo–san
- Treebankers: Maruyama–san; Morimoto–san
- Consultant: Tagaki–san

Crucial to our work has been the assistance of the administrative staff of ITL, ATR, and ATRI, whom we wish to thank enthusiastically.

Finally, our research has been supported by our management, throughout the life of the Statistical Parsing Group. We wish to express our gratitude to our management for this support, and for their enthusiasm and their assistance in myriad ways:

- Supervisors: Sumita–san (93–94); Kashioka–san (97–present)
- Department 3 Managers: Iida–san (93–98); Shirai–san (98–present)
- ITL Directors: Yamazaki–san (93–97); Yamamoto–san (97–present)

2 Overview

The goal of the Statistical Parsing Group has been, and continues to be, the development of an accurate and fast parser of unrestricted English text which supplies grammatical analyses that are extremely detailed both syntactically and semantically. Our conviction has been that only such thoroughgoing linguistic analyses have a chance of being genuinely useful over the entire spectrum of language–based applications within Artificial Intelligence. To be as useful as possible to this set of applications is the purpose of the work of our group.

The present report will describe and detail the actions we have taken in our attempt to realize our goal of producing a fast and accurate parser for unlimited–domain, unrestricted English text. We have not fully realized this aim as yet, but we feel we are within sight of it. Further, we have quantified the degree of help to various Artificial Intelligence applications (speech recognition and speech synthesis) accruing from the linguistic analyses of our system, for the hypothetical case in which we are always able to deliver the correct analysis, with respect to our grammar. We believe that the magnitude of this help is such as to amply justify our continuing effort to achieve a very high degree of accuracy in predicting the correct grammatical analysis, with respect to our grammar, for any input sentence of English.

The organization of this report is as follows: Section 3 describes our approach to linguistically annotating English text, and in particular discusses the roughly–1–million–word training database we have constructed in order to supply a large number of examples of correctly analyzed English text to our machine–learning, predictive algorithms. Section 4 describes the first of two predictive devices which we have constructed with the aim of supplying, for any input sentence of English, the correct grammatical analysis as implied by our system of linguistic annotation of text: this is our part–of–speech tagger, whose function is to associate with any string of English words a sequence of labels, one per word, which identify the syntactic and semantic functioning of that word in the sentence in which it occurs. Section 5 gives an account of our parser, which is the second of the two predictive devices mentioned just above. Its function is to associate with a part–of–speech–tagged sentence of English, its correct grammatical (syntactic and semantic) structure, as implied by our system of linguistic annotation of text (our grammar).

Section 6 of this report presents the results of applying our system of linguistic annotation of text to two different, classic, language–based tasks within Artificial Intelligence: speech recognition and speech synthesis. Together, the two sets of results we present serve to illustrate the quite considerable value which can be provided to typical Artificial Intelligence applications when the detailed and exacting level of linguistic analysis which our annotation scheme delivers, is projected accurately onto new and previously unknown sentences of English with which some particular application finds it has to deal. This advantage in turn represents a challenge for the future work of the Statistical Parsing Group: the challenge is to so hone the predictive capacities of our machine–learning component, that we are able reliable and quickly, to locate the correct parse, the correct tag, for a given sentence or word in the context in which it has occurred. This is the task our group confronts in its continuation under the new research program which is to be the successor to the Interpreting Telecommunications Laboratory: to take our success at annotating language in a manner full enough of "information" to yield substantial improvements in the functioning of a range of typical language–based Artificial Intelligence tasks—to take that success and build on it by enhancing our ability to find the exactly correct analysis for a sentence or word, among many possible such analyses available from the grammar, up to a point where the potential improvements in Artificial Intelligence tasks which we have already deomnstrated, can be realized in action.

3 Providing Training Data: Annotation System for English Text

Note: This section is taken from an article presented in 1996, so that occasional wording reflects this out–of–dateness. However, notes within the text have been added to update information where necessary (EB 1999).

3.1 Introduction

A *treebank* is a body of natural–language text which has been grammatically annotated by hand, in terms of some previously–established scheme of grammatical analysis. Treebanks have been used within the field of natural–language processing as a source of training data for statistical part–of–speech taggers (Black et al., 1992; Brill, 1994; Garside et al., 1987; Merialdo, 1994; Weischedel et al., 1993) and for statistical parsers (Black et al., 1993; Brill, 1993; Garside et al., 1987; Jelinek et al., 1994; Magerman, 1995; Magerman and Marcus, 1991; Sekine and Grishman, 1995).

In this section, we present the *ATR/Lancaster Treebank of American English*, a resource for natural–language–processing research, which has been prepared by Lancaster University (UK)'s Unit for Computer Research on the English Language, according to specifications provided by ATR (Japan)'s Statistical Parsing Group. First we provide a "static" description, with (a) a discussion of the mode of selection and initial processing of text for inclusion in the treebank, and (b) an explanation of the scheme of grammatical annotation we then apply to the text. Second, we supply a "process" description of the treebank, in which we detail the physical and computational mechanisms by which we have created it. Finally, we lay out plans for the further development of this new treebank.

All of the features of the ATR/Lancaster Treebank that are described below represent a radical departure from extant large–scale (Eyes and Leech, 1993; Garside and McEnery, 1993; Marcus et al., 1993) and smaller-scale (Sampson, 1994; van Halteren and van den Heuvel, 1990) treebanks. We have chosen in this section to present our treebank in some detail, rather than to compare and contrast it with other treebanks. But the major differences between this and earlier treebanks can easily be grasped via a comparison of the descriptions below with those of the sources just cited.

3.2 General Description of the Treebank

3.2.1 Document Selection and Preprocessing

The ATR/Lancaster Treebank consists of approximately 500,000 words[1] of grammatically–analyzed text divided into roughly 650 documents ranging in length from about 30 to about 3600 words.

The idea informing the selection of documents for inclusion in this new treebank was to pack into it the maximum degree of document variation along many different scales—document length, subject area, style, point of view, etc.—but without establishing a single, predetermined classification of the included documents.[2] Differing purposes for which the treebank might be utilized may favor differing groupings or classifications of its component documents. Overall, the rationale for seeking to take as broad as possible a sample of current standard American English, is to support the parsing and tagging of unconstrained American English text by providing a training corpus which includes documents fairly similar to almost any input which might arise.

Documents were obtained from three sources: the Internet; optically–scanned hardcopy "occasional" documents (restaurant take–out menus; fundraising letters; utility bills); and purchase

[1] this reached 700,000 by Spring 1996, and ultimately 1.1 million words (EB 1999)

[2] as was done, by contrast, in the Brown Corpus (Kucera and Francis, 1967).

Empire Szechuan Flier (Chinese take–out food)
Catalog of Guitar Dealer
UN Charter: Chapters 1–5
Airplane Exit–Row Seating: Passenger Information Sheet
Bicycles: How To Trackstand
Government: US Goals at G7
Shoe Store Sale Flier
Hair–Loss Remedy Brochure
Cancer: Ewing's Sarcoma Patient Information

Table 1: Nine Typical Documents From ATR/Lancaster Treebank

from commercial or academic vendors. To illustrate the diverse nature of the documents included in this treebank, we list, in Table 1, titles of nine typical documents.

In general, and as one might expect, the documents we have used were written in the early to mid 1990s, in the United States, in "Standard" American English. However, there are fairly many exceptions: documents written by Captain John Smith of Plymouth Plantation (1600s), by Benjamin Franklin (1700s), by Americans writing in periods throughout the 1800s and 1900s; documents written in Australian, British, Canadian, and Indian English; and documents featuring a range of dialects and regional varieties of current American English. A smattering of such documents is included because within standard English, these linguistic varieties are sometimes quoted or otherwise utilized, and so they should be represented.

As noted above, each document within the treebank is classified along many different axes, in order to support a large variety of different task–specific groupings of the documents. Each document is classifed according to TONE, STYLE, LINGUISTIC.LEVEL, POINT.OF.VIEW, PHYSICAL.DESCRIPTION.OF.DOCUMENT, GEOGRAPHICAL.BACKGROUND.OF.AUTHOR, etc. Sample values for these attributes are: "friendly", "dense", "literary", "technical", "how-to guide", and "American South", respectively. To convey domain information, one or more Dewey Decimal System three–digit classifiers are associated with each document. For instance, for the cv of a physiologist, Dewey 612 and 616 (Medical Sciences: Human Physiology; Diseases) were chosen. On a more mundane, "bookkeeping" level, values for TEXT.TITLE, AUTHOR, PUBLICATION.DATE, TEXT.SOURCE, etc. are recorded as well.

An SGML–like markup language is used to capture a variety of organizational–level facts about each document, such as LIST structure; TITLEs and CAPTIONs; and even more recondite events such as POEM and IMAGE. HIGHLIGHTing of words and phrases is recorded, along with the variety of highlighting: italics, boldface, large font, etc. Spelling errors and, where essential, other typographical lapses, are scrupulously recorded and then corrected.

Tokenization (i.e. word–splitting: Edward's $\longrightarrow$ Edward 's) and sentence–splitting (e.g. He said, "Hi there. Long time no see." $\longrightarrow$ (Sentence.1:) He said, (Sentence.2:) "Hi there. (Sentence.3:) Long time no see.") are performed by hand according to predetermined policies. Hence the treebank provides the resource of multifarious correct instances of word– and sentence–splitting.

3.2.2 Scheme of Grammatical Annotation

Heretofore, all existing large–scale treebanks have employed the grammatical analysis technique of *skeleton parsing* (Eyes and Leech, 1993; Garside and McEnery, 1993; Marcus et al., 1993), in which only a partial, relatively sketchy, grammatical analysis of each sentence in the treebank is provided.[3][4] In contrast, the ATR/Lancaster Treebank assigns to each of its sentences a full and complete grammatical analysis with respect to a very detailed, very comprehensive broad–coverage grammar of English. Moreover, a very large, highly detailed part–of–speech tagset is used to label each word of each sentence with its syntactic *and* semantic categories. The result is an extremely specific and informative syntactic and semantic diagram of every sentence in the treebank.

This shift from skeleton–parsing–based treebanks to a treebank providing full, detailed grammatical analysis resolves a set of problems, detailed in (Black, 1994a), involved in using skeleton–parsing–based treebanks as a means of initializing training statistics for probabilistic grammars (Black et al., 1993). Briefly, the first of these problems, which applies even where the grammar being trained has been induced from the training treebank (Sharman et al., 1990), is that the syntactic sketchiness of a skeleton–parsed treebank leads a statistical training algorithm to overcount, in some circumstances, and in other cases to undercount instances of rule firings in training–data (treebank) parses, and thus to incorrectly estimate rule probabilities. The second problem is that where the grammar being trained is more detailed syntactically than the skeleton–parsing–based training treebank, the training corpus radically underperforms in its crucial job of specifying correct parses for training purposes (Black, 1994a).

In addition to resolving grammar–training problems, our Treebank provides a means of training non–grammar–based parsing procedures (Brill, 1993; Jelinek et al., 1994; Magerman, 1995) at new, higher levels of grammatical detail and comprehensiveness.

Treebank sentences are parsed in terms of the *ATR English Grammar*, whose characteristics we will briefly describe.

The Grammar's part–of–speech tagset resembles the 179–tag Claws tagset developed by UCREL (Eyes and Leech, 1993), but with numerous major and minor differences. One major difference, for instance, is that the ATR tagset captures the difference between e.g. "wall covering", where "covering" is a lexicalized noun ending in -ing, and "the covering of all bets", where "covering" is a gerund. In Claws practice, both are NN1 (singular common noun). The ATR tagset innovates the tag type NVVG for gerunds. Another major difference is the (limited) use of "subcategorization", e.g. VDBLOBJ for double–object verbs (teach Bill Latin, etc.).

Each verb, noun, adjective and adverb in the ATR tagset includes a semantic label, chosen from 42 noun/adjective/adverb categories and 29 verb/verbal categrories, some overlap existing between these category sets. These semantic categories are intended for *any "Standard–American–English" text, in any domain.* Sample categories include: "physical.attribute" (nouns/adjectives/adverbs), "alter" (verbs/verbals), and "interpersonal.act" (nouns/adjectives/adverbs/verbs/verbals). They were developed by ATR staff and then proven and refined via day–in–day–out tagging for six months at ATR by two human "treebankers", then for four months at Lancaster by five treebankers, with daily interactions among treebankers, and between the treebankers and ATR staff.

[3]The 1995–release Penn Treebank adds functional information to some nonterminals (Marcus et al., 1994), but with its rudimentary (roughly–45–tag) tagset, its non–detailed internal analysis of noun compounds and NPs more generally, its lack of semantic categorization of words and phrases, etc., it arguably remains a skeleton–parsed treebank, albeit an enriched one.

[4]A different kind of partial parse—crucially, one generated automatically and not by hand—characterizes the "treebank" produced by processing the 200–million–word Birmingham University (UK) Bank–of–English text corpus with the dependency–grammar–based ENGCG Helsinki Parser (Karlsson et al., 1995).

```
<S id="20" count=13>
<HIGH rendition ="italic">
[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]
</HIGH>
</S>
```

Figure 1: *ATR/Lancaster Treebank* Sentence (wordlength: 13; sentence italicized) from Credit Union Brochure

If we ignore the semantic portion of ATR tags, the tagset contains 165 different tags. Including the semantic categories in the tags, there are roughly 2200 tags. As is the case in the Claws tagset, so–called "ditto tags" can be created based on almost any tag of the tagset, for the purpose of labelling multiword expressions. For instance, "will o' the wisp" is labelled as a 4–word singular common noun. This process can add considerable numbers of tags to the above totals.

Sentences in the Treebank are parsed with respect to the *ATR English Grammar*. The Grammar, a feature–based context–free phrase–structure grammar, is related to the IBM English Grammar as published in (Black et al., 1993), but differs more from the IBM Grammar than our tagset does from the Claws tagset. For instance, the notion of "mnemonic" has no application to the ATR Grammar; the ATR Grammar has 67 features and 1100 rules, whereas the IBM Grammar had 40 features and 750 rules, etc.

The precisely correct parse (as pre–specified by a human "treebanker") figures *among* the parses produced for any given sentence by the ATR Grammar, roughly 90% of the time, for text of the unconstrained, wide–open sort that the Treebank is composed of. The job of the treebankers is to locate this exact parse, for each sentence, and add it to the Treebank.

Figure 1 shows a sample parsed sentence from the ATR Treebank. Because it is informative to know which of the 1100 rules is used at a given tree node, and since the particular "nonterminal category" associated with any node of the tree is always recoverable,[5] nodes are labelled with ATR Grammar rule names rather than, as is more usual, with nonterminal names. Figure 2 shows two Treebank sentences from a Chinese take–out food flier.

3.3 Producing the Treebank

In this part of the article, we turn from "what" to "how", and discuss the mechanisms by which the ATR/Lancaster Treebank was produced.

[5] It is contained in the rule name itself.

```
<S id="39" count=8>
<HIGH rendition="italic">
[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]
</HIGH>
</S>

<S id="48" count=5>
<HIGH rendition ="large">
[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]
</HIGH>
</S>
```

Figure 2: *Two ATR/Lancaster Treebank* Sentences (8 words, italicized; 5 words, large font) from Chinese Take–Out Food Flier

3.3.1 The Software Backbone—GWBTool: A Treebanker's Workstation

GWBTool is a Motif-based X-Windows application which allows the treebanker to interact with the ATR English Grammar in order to produce the most accurate treebank in the shortest amount of time.

The treebanking process begins in the Treebank Editor screen of the treebanker's workstation with a list of sentences tagged with part-of-speech categories. The treebanker selects a sentence from the list, for processing. Next, with the click of a button, the Treebank Editor graphically displays the parse forest for the sentence in a mouse-sensitive Parse Tree window (Figure 3). Each node displayed represents a constituent in the parse forest. A shaded constituent node indicates that there are alternative analyses of that constituent, only one of which is displayed. By clicking the right mouse button on a shaded node, the treebanker can display a popup menu listing the alternative analyses, any of which can be displayed by selecting the appropriate menu item. Clicking the left mouse button on a constituent node pops up a window listing the feature values for that constituent.

The Treebank Editor also displays the number of parses in the parse forest. If the parse forest is unmanageably large, the treebanker can partially bracket the sentence and, again with the click of a button, see the parse forest containing only those parses which are consistent with the partial bracketing (i.e. which do not have any constituents which violate the constituent boundaries in the partial bracketing). Note that the treebanker need not specify any labels in the partial bracketing, only constituent boundaries. The process described above is repeated until the treebanker can narrow the parse forest down to a single correct parse. Crucially, for experienced Lancaster treebankers, the number of such iterations is, by now, normally none or one.

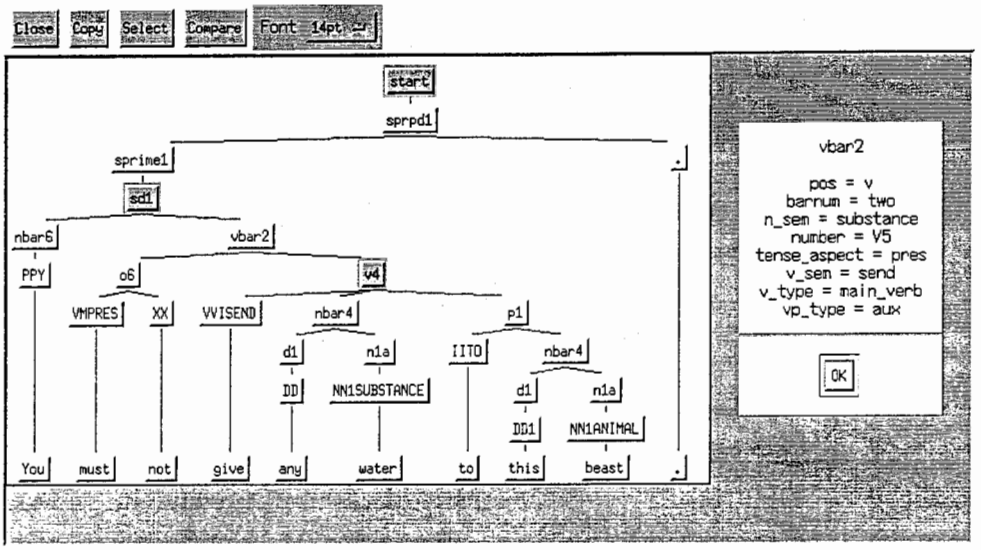

Figure 3: The GWBTool Treebanker's Workstation Parse Window display, showing the parse forest for an example sentence. On the far right, the feature values of the VBAR2 constituent, indicating that the constituent is an auxiliary verb phrase (bar level 2) containing a present–tense verb phrase with noun semantics SUBSTANCE and verb semantics SEND. The fact that the number feature is variable (NUMBER=V5) indicates that the number of the verb phrase is not specified by the sentence. The shaded nodes indicate where there are alternative parses.

3.3.2 Two–Stage Part–Of–Speech Tagging

Part-of-speech tags are assigned in a two-stage process: (a) one or more potential tags are assigned automatically using the Claws HMM tagger (Garside et al., 1987); (b) the tags are corrected by a treebanker using a special-purpose X-windows-based editor, Xanthippe. This displays a text segment and, for each word contained therein, a ranked list of suggested tags. The analyst can choose among these tags or, by clicking on a panel of all possible tags, insert a tag not in the ranked list.

The automatic tagger inserts only the syntactic part of the tag. To insert the semantic part of the tag, Xanthippe presents a panel representing all possible semantic continuations of the syntactic part of the tag selected.

Tokenization, sentence–splitting, and spell–checking are carried out according to rule by the treebankers themselves (see 2.1 above). However, the Claws tagger performs basic and preliminary tokenization and sentence–splitting, for optional correction using the Xanthippe editor. Xanthippe retains control at all times during the tag correction process, for instance allowing the insertion only of tags valid according to the ATR Grammar.

3.3.3 The Annotation Process

Initially a file consists of a header detailing the file name, TEXT.TITLE, AUTHOR, etc., and the text itself, which may be in a variety of formats; it may contain HTML mark-up, and files vary in the way in which, for example, emphasis is represented. The first stage of processing is a scan of the text to establish its format and, for large files, to delimit a sample to be annotated.

The second stage is the insertion of SGML–like mark-up. As with the tagging process, this is done by an automatic procedure with manual correction, using microemacs with a special set of macros.

Third, the tagging process described in section 3.2 is carried out. The tagged text is then extracted into a file for parsing via GWBTool (See 3.1.1).

The final stage is merging the parsed and tagged text with all the annotation (SGML–like mark–up, header information) for return to ATR.

3.3.4 Staff Training; Output Accuracy

Even though all Treebank parses are guaranteed to be acceptable to the ATR Grammar, insuring consistency and accuracy of output has required considerable planning and effort. Of all the parses output for a sentence being treebanked, only a small subset are appropriate choices, given the sentence's meaning in the document in which it occurs. The five Lancaster treebankers had to undergo extensive training over a long period, to understand the manifold devices of the ATR Grammar expertly enough to make the requisite choices.

This training was effected in three ways: a week of classroom training was followed by four months of daily email interaction between the treebankers and the creator of the ATR Grammar; and once this training period ended, daily Lancaster/ATR email interaction continued, as well as constant consultation among the treebankers themselves. A body of documentation and lore was developed and frequently referred to, concerning how all semantic and certain syntactic aspects of the tagset, as well as various grammar rules, are to be applied and interpreted. (This material is organized via a menu system, and updated at least weekly.) A searchable version of files annotated to date, and a list of past tagging decisions, ordered by word and by tag, are at the treebankers' disposal.

In addition to the constant dialogue between the treebankers and the ATR grammarian, Lancaster output was sampled periodically at ATR, hand–corrected, and sent back to the treebankers. In this way, quality control, determination of output accuracy, and consistency control were handled conjointly via the twin methods of sample correction and constant treebanker/grammarian dialogue.

With regard both to accuracy and consistency of output analyses, individual treebanker abilities clustered in a fortunate manner. Scoring of thousands of words of sample data over time revealed that three of the five treebankers had parsing error rates (percentage of sentences parsed incorrectly) of 7%, 10%, and 14% respectively, while the other two treebankers' error rates were 30% and 36% respectively. Tagging error rates (percentage of all tags that were incorrect), similarly, were 2.3%, 1.7%, 4.0%, 7.3% and 3.6%. Expected parsing error rate worked out to 11.9% for the first three, but 32.0% for the other two treebankers; while expected tagging error rates were 2.9% and 6.1% respectively.[6]

What is fortunate about this clustering of abilities is that the less able treebankers were also much less prolific than the others, producing only 30% of the total treebank. Therefore, we are provisionally excluding this 30% of the treebank (currently about 150,000 words) from use for parser training, though we are experimenting with the use of the entire treebank (expected tagging error rate: 3.9%) for tagger training. Finally, parsing and tagging consistency among the first three treebankers appears high.

3.4 Conclusion

Within the next two years, we intend to produce Version 2 of our Treebank, in which the 30% of the treebank that is currently suitable for training taggers but not parsers, is fully corrected.[7]

Over the next several years, the ATR/Lancaster Treebank of American English will form the basis for the research of ATR's Statistical Parsing Group in statistical parsing, part–of–speech tagging, and related fields.[8]

4 Predicting Word Meaning and Function: Part–Of–Speech Tagging

Note: Some of the performance results in this section are slightly dated, taken as the section is from work published in 1997. In particular, overall tagging results on our "golden–standard" test set (see Section 5) are currently around 85%. Further, the article on which this section is based partakes of a somewhat polemical orientation, which is best overlooked in the current context as being beside the point.

4.1 Introduction: Building On The Successes To Date In Part–Of–Speech Tagging

Part–of–speech tagging—using computers to automatically associate the words of a text with their grammatical parts of speech—has been one of the success stories of the Natural Language Processing

[6] Almost all tagging errors were semantic.

[7] Seven–tenths of this 30% is already correct, so that the task involved is re–parsing 30% of 30% (= 9%) of the treebank. Further note (EB 1999): the correction of this 30% of the treebank was in fact finally begun in late 1998, and is ongoing as of this writing.

[8] This is in fact what happened (EB 1999).

Empire Szechuan Flier (Chinese take–out food)
Catalog of Guitar Dealer
UN Charter: Chapters 1–5
Airplane Exit–Row Seating: Passenger Information Sheet
Bicycles: How To Trackstand
Government: US Goals at G7
Shoe Store Sale Flier
Hair–Loss Remedy Brochure
Cancer: Ewing's Sarcoma Patient Information

Table 2: Nine Typical Documents From ATR/Lancaster Treebank

field to date. Computers have equalled human accuracy at tagging the Wall Street Journal, Brown Corpus, Associated Press, and Canadian Hansard corpora,[9] using the rudimentary, 45–tag UPenn Tagset,[10] and stripped–down versions of the fuller CLAWS tagset.[11]

But what will the ability to tag with these relatively low–level tagsets do for complex applications such as machine translation, sophisticated document–searching, and open–vocabulary speech recognition? The logical next move for part–of–speech tagging is to build on its successes and undertake more complex and challenging tagging tasks.

Three directions for expansion seem indicated: (1) tag using much more detailed tagsets, including a large-scale semantic classification as well as more syntactic detail; (2) test performance on treebanks which reflect the huge gamut of domains, styles, functions, and usages found among real–world applications; and (3) understand the magnitude of the unknown–word and unknown–tag problems, then overcome them.

One way to confront all these problems is to tag using the 700,000–word[12] *ATR/Lancaster Treebank of American English* (Black et al., 1996). Divided into roughly 950 documents of length 30–3600 words, this treebank achieves a high degree of document variation along many different scales—document length, subject area, style, point of view, etc. (See Table 1 for titles of nine typical documents.) Text is tagged and parsed using the *ATR English Grammar* (2720 different tags). Each verb, noun, adjective and adverb tag includes one of about 60 semantic categories intended for any Standard American English text in any domain. Even the syntax–only version of the tagset has 443 different tags. (Compare 45, 76, and 163 tags for the tagsets used in (Brill, 1994; Weischedel et al., 1993; Merialdo, 1994; Black et al., 1992).) The unknown–word and unknown–tag problems[13] are quantified below and turn out to be much more severe than one might have thought. Unknown–tag difficulties are sufficiently acute in ATR/Lancaster–Treebank test sets to form a spur to solving the problem.

[9] (Brill, 1994; Weischedel et al., 1993; Merialdo, 1994; Black et al., 1992; Marcus et al., 1993; Kucera and Francis, 1967; Garside and McEnery, 1993)

[10] (Marcus et al., 1993)

[11] (Eyes and Leech, 1993; Garside et al., 1987; Garside and McEnery, 1993)

[12] now 1.1–million–word (EB 1999)

[13] viz., the word to be tagged (a) has never been encountered in the training corpus (unknown–word); or (b) is in the training corpus, but not with the tag which it needs to be assigned in the case at hand (unknown–tag)

4.2 Real–World Part–Of–Speech Tagging

In Section 2, we document the problems of tagging with larger, more sophisticated tagsets (2.1), and of tagging unknown words and words occurring with a given tag for the first time in test data (2.2), and show why it is important to solve these problems. Section 3 describes the solution we are attempting, using decision–tree modelling and discarding the notion of a dictionary entirely (3.1); and presents experimental results and future research plans (3.2).

4.2.1 Tag Using Tagsets Of Increased Size And Complexity

This subsection seeks to convey a "feel" for the increasing levels of detail of the tagsets utilized so far in tagging work—including the new ATR tagsets. Then, specific cases are discussed of syntactic details captured in the ATR tagset but not in tagsets used for prior tagging experiments, and it is shown why these details matter.

Exemplifying The Tagsets Used So Far Tables 2–5 display the full text of a 1989 Wall Street Journal article entitled, "Enserch Tender Offer Results", tagged using first the full 2720–tag ATR tagset (*ATR–Full*); then the 443–tag syntax–only version of the ATR tagset (*ATR–Syntax*); then the 163–tag mapped–down version of the CLAWS tagset which was used in (Black et al., 1992); then finally the 45–tag UPenn tagset used in (Brill, 1994).

(See footnotes for glosses of ATR–Full, ATR–Syntax,[14] mapped–down CLAWS,[15] and UPenn[16] tagged versions.)

[14]Gloss (ATR–Full in italics; ATR–Syntax in boldface); tags common to both in boldface: . period; , comma; **APP$** possessive pronoun, pre–nominal: my, our; **AT** article, either singular or plural: the, no; **AT1** singular article: a, every; **CC** coordinating conjunction; *CCAND* "and"; *CCOR* "or"; **CSN** "than" as conjunction: nicer than I thought; **CST** "that" as conjunction: that he is here; **DAR** comparative after–determiner: more, less; **DB** before–determiner: all, half; **DD1** singular determiner: this, another; **II** preposition; *IIFROM* "from"; *IIOF* "of"; *IION* "on"; *IITO* "to"; **JJVVN** past participle used as adjective; *JJVVNINTER-ACT* "inter_action": the deposed Shah, the stolen car; *JJVVNCONTROL* "control": unsecured notes, management–led buy–out; **MC** digital cardinal number: 2, 3; **MCWORD21, MCWORD22** two–part cardinal number, in words: six hundred, three dozen; **MDATEWORD** date, in words: Monday, April; **NNUNUM** number followed by unit of measurement: 6cc, 8in.; **NN1** singular common noun; *NN1COMP–B* "complex_behavior": bankruptcy, research; *NN1MONEY* "money": grant, fine; *NN1PERS–ATT* "personal_attribute": ability, nose; *NN1SYSTEM-PT* "system_part": cabinet (meeting), precinct (caucuses); *NN1VERBAL–ACT* "verbal_act": (the) claim, revelation; **NN2** plural common noun; *NN2ABS–UNIT* "abstract_unit": alternates, breaks; **NP1** singular proper noun; *NP1CITYNM* "cityname": Toronto; *NP1FRMNM* "firmname": GE, Hitachi; *NP1INSTIT* "instit": School, Club; *NP1INSTITNM* "institname": Harvard, 4–H; *NP1POSTFRMNM* "postfirmnname": Inc., Ltd.; *NP1PREPLCNM* "preplacename": St. (Louis), Los (Alamos); **RR** general adverb; *RRDEGREE* "degree": absolutely, approximately; *RRINTER-ACT* "inter_action": jointly, closely; **TO** pre–infinitival element: to (walk), to (go); **VBDR** were; **VBI** infinitive form of verb "be": be; **VMPRES** "present" modal auxiliary: can, will; **VVD** simple past verb; *VSAYINGD* "saying": claimed, stated; *VVDINCHOATIVE* "inchoative": achieved, created; *VVDVERBAL-ACTSD* "verbal-act", takes sentential complement: implied, mentioned; **VVI** infinitive verb; *VVIALTER* "alter": adjust, slacken; *VVIPROCESSIVE* "processive": continue, break; **VVN** past participle; *VVNINTER–ACT* "inter_action": (It was) sold, (They were) arrested.

[15]Gloss (non–obvious tags only): **NNJ** organization noun, neutral for number; **NNL1** singular locative noun; **NNO** numeral noun, neutral for number; **NNU** unit of measurement, neutral for number; **NPD1** singular weekday noun; **RG** degree adverb; (NB: "in response to" would be tagged II31,II32,II33 (three–word preposition), using the CLAWS tagset, of which the present tagset is a mapped–down version).

[16]Gloss (non–obvious tags only): **CD** Cardinal number; **IN** Preposition or subordinating conjunction; **MD** Modal; **NN** Noun, singular or mass; **NNP** Proper noun, singular; **NNS** Noun, plural; **PRP$** Possessive pronoun; **RB** Adverb; **TO** to; **VB** Verb, base form; **WDT** Wh–determiner.

Table 3: Sentence 1:

WORD	ATR	Lanc	Upenn
Enserch	NP1(FRMNM)	NP1	NNP
Corp.	NP1(POSTFRMNM)	NNJ	NNP
said	VVD(VERBAL-ACTSD)	VVD	VBD
about	RR(DEGREE)	RG	RB
12	MCWORD21	MC	CD
million	MCWORD22	NNO	CD
,	,	,	,
or	CC(OR)	CC	CC
93			CD
%	NNUNUM	NNU	NN
,	,	,	,
of	II(OF)	IO	IN
the	AT	AT	DT
publicly	RR(INTER-ACT)	RR	RB
traded	JJVVN(INTER-ACT)	VVN	VBN
units	NN2(ABS-UNIT)	NN2	NNS
of	II(OF)	IO	IN
its	APP$	APP$	PRP$
limited	JJVVN(CONTROL)	JJ	JJ
partnership	NN1(SYSTEM-PT)	NN1	NN
,	,	,	,
Enserch	NP1(FRMNM)	NP1	NNP
Exploration	NN1(COMP-B)	NN1	NNP
Partners	NP1(POSTFRMNM)	NN2	NNP
Ltd.	NP1(POSTFRMNM)	JJ	NNP
,	,	,	,
were	VBDR	VBDR	VBD
tendered	VVN(INTER-ACT)	VVN	VBN
in	II(IN)	II	IN
response	NN1(VERBAL-ACT)	II	NN
to	II(TO)	II	TO
an	AT1	AT1	DT
offer	NN1(INTER-ACT)	NN1	NN
that	CST	CST	WDT
expired	VVD(INCHOATIVE)	VVD	VBD
Monday	MDATEWORD	NPD1	NNP
.	.	.	.

Table 4: Sentence 2:

WORD	ATR	Lanc	Upenn
Enserch	NP1(FRMNM)	NP1	NNP
said	VVD(VERBAL-ACTSD)	VVD	VBD
the	AT	AT	DT
tendered	JJVVN(INTER-ACT)	JJ	VBN
units	NN2(ABS-UNIT)	NN2	NNS
will	VMPRES	VM	MD
raise	VVI(ALTER)	VVI	VB
its	APP$	APP$	PRP$
ownership	NN1(PERS-ATT)	NN1	NN
of	II(OF)	IO	IN
the	AT	AT	DT
partnership	NN1(SYSTEM-PT)	NN1	NN
to	II(TO)	II	TO
more	DAR	DAR	JJR
than	CSN	CSN	IN
99			CD
%	NNUNUM	NNU	NN
from	II(FROM)	II	IN
87			CD
%	NNUNUM	NNU	NN
.	.	.	.

Table 5: Sentence 3:

WORD	ATR	Lanc	Upenn
About	RR(DEGREE)	RG	RB
900,000	MC	MC	CD
units	NN2(ABS-UNIT)	NN2	NNS
will	VMPRES	VM	MD
continue	VVI(PROCESSIVE)	VVI	VB
to	TO	TO	TO
be	VBI	VBI	VB
publicly	RR(INTER-ACT)	RR	RB
traded	VVN(INTER-ACT)	VVN	VBN
on	II(ON)	II	IN
the	AT	AT	DT
New	NP1(PREPLCNM)	NP1	NNP
York	NP1(CITYNM)	NP1	NNP
Stock	NN1(MONEY)	NN1	NNP
Exchange	NP1(INSTIT)	NNL1	NNP
,	,	,	,
Enserch	NP1(FRMNM)	NP1	NNP
said	VVD(SAYING)	VVD	VBD
.	.	.	.

Table 6: Sentence 4:

WORD	ATR	Lanc	Upenn
Enserch	NP1(FRMNM)	NP1	NNP
had	VHD	VHD	VBD
offered	VVN(INTER-ACT)	VVN	VBN
one-half	DB	DB	NN
a	AT1	AT1	DT
share	NN1(ABS-UNIT)	NN1	NN
of	II(OF)	IO	IN
its	APP$	APP$	PRP$
common	NN1(MONEY)	NN1	JJ
and	CC(AND)	CC	CC
$			$
1	MPRICE	NNU	CD
in	II(IN)	II	IN
cash	NN1(MONEY)	NN1	NN
for	II(FOR)	IF	IN
each	DD1	DD1	DT
unit	NN1(ABS-UNIT)	NN1	NN
.	.	.	.

WSJ Article "Enserch Tender Offer Results", Tagged Using ATR–Full, ATR–Syntax, Mapped–Down CLAWS, And UPenn Tagsets.

Within "ATR" column, portions of a tag present in ATR–Full but not ATR–Syntax are parenthesized.

Note tokenization (word–splitting) differences between ATR, CLAWS on one hand, and UPenn on other: 99% and $1 are one word for the former tagsets, two words for the latter.

Expanded Syntactic Detail: The ATR–Syntax Tagset The ATR–Syntax tagset is a revised and expanded version of the CLAWS tagset. Three areas of ATR–Syntax's increased syntactic detail vis–a–vis previously–utilized tagsets are now discussed.

Ditto Tags For Multiword Lexical Units The CLAWS tagset features so–called *ditto tags* for multiword lexical units. For instance, well_NN121 being_NN122 is a two–word noun, in_II31 response_II32 to_II33 a three-word preposition.[17] Many more ditto tags are contained in ATR–Syntax than in CLAWS, with 276 of the 443 ATR–Syntax tags being ditto tags.[18]

All CLAWS ditto tags are mapped out of both the (Merialdo, 1994) and (Black et al., 1992) experiments, so that no published experiments have appeared to date with tagsets featuring ditto tags. In (Black et al., 1992), the "ditto endings" are dropped, so that e.g. well_NN121 being_NN122 becomes well_NN1 being_NN1. It is not clear how ditto tags were handled in (Merialdo, 1994); in any case, the full mapped–down 76–tag tagset is exhibited in (Merialdo, 1994), and no ditto tags are included.

What is the advantage of marking certain multiword lexical units, and why is it more useful to have explicit ditto tags than mapped–down ones as in (Black et al., 1992)? One answer concerns what happens when one runs a parser[19] on tagged text. Briefly, tagging e.g. in_II31 response_II32 to_II33, tells the parser to treat the three words as a single preposition, and so to ignore possible breakdowns like: "He nodded (in response) (to show he was following)", of the the sentence containing the phrase when so tagged. This can turn out to be a significant aid to parsing accuracy, if ditto tags appear frequently in correctly–tagged text, since large numbers of mistaken parses are eliminated which might otherwise be considered correct by the parser.

Dropping the "ditto" sequence markers, i.e. mapping the above to in_II response_II to_II, as was done in (Black et al., 1992), goes part–way towards the above goal, in that it prevents parsing mistakes like the one above. But it does nothing to block errors such as partitioning the phrase, "the comments he made in response to this question" as if it were of the form, "the options he chose from among (in this case)" or, "the option he chose from (among (in this case) five possibilities)", etc.

An extremely frequent and potentially havoc–wreaking ditto–tag scenario occurs where a multiword adverb occurs at the end of a sentence, especially a long sentence. Locutions like, "as_RR21 well_RR21", "a_RR21 lot_RR22", "by_RR31 and_RR32 large_RR33" are common in this position. If we denature the ditto tags into a series of two or three adverbs, the number of otherwise–preventable spurious parses now open to an unsuspecting parser can be huge. Even among short sentences there are many variations: He (paid ($5000 precisely) (wholly willingly)); (He (paid ($5000 precisely) unhesitatingly) sometimes); (He (spent money (terribly freely)) always); etc.

Digit–Based And Number–Word–Based Lexical Units: Price, Time, Etc. Among the 276 ditto tags in the ATR tagset, 170 are for digit–based or number–word–based lexical units, e.g. MPRICE31, MTIMEWORD22, MZIP21. In addition, the set of standard (non–ditto–tag)

[17] Obviously, only appropriate occurrences of these word sequences are tagged as above. E.g., "He nodded in response to indicate he was listening" would not be so tagged.

[18] Since this leaves only 167 tags, it might be thought that apart from ditto tags, the CLAWS and ATR–Syntax tagsets are the same. This is far from true, however, since 39 of these 167 non–ditto ATR–Syntax tags are absent from the CLAWS tagsets, and 51 of the 179 tags of the CLAWS2a tagset, for instance, (Eyes and Leech, 1993), do not figure in the ATR–Syntax tagset. Broad discussion of these differences is beyond the scope of this article, but two important cases will be dealt with in the next two subsections.

[19] a device for automatically diagramming sentences

ATR tags contains 21 other tags for single–word lexical units of this type. All 191 of these tags are identical for the ATR–Full and ATR–Syntax tagsets. That is, in both, a full panoply of tags for prices, times, zipcodes, and the like, is included, along with a variety of tags for "just plain numbers", e.g. fifty_MCWORD21 three_MCWORD22, 1_MC1, next_MDWORD, 325–92_MC-MC (e.g. a 325–92 vote). The rationale here is that it is feasible for a tagger to learn to demarcate multiword price, time, zipcode, etc., expressions, and that specifying the internal structure of these expressions is probably of lesser utility in general. What *is* quite important is to locate the boundaries of these wordstrings, which often include highly frequent words which if not rendered harmless in this fashion, might encourage significant numbers of misparses. For instance, the "a" of "a hundred fifty", the "the" of "Tuesday the 19th", and the "bits" of "two bits",[20] need to be identified as occurring inside numerical lexical items, if they are not to sow confusion.

Are these tags "syntactic"? Merely to pose this question suggests a need for at least ten years of "Wittgensteinian therapy". If anyone wishes, we can change the name "ATR–Syntax tagset" to "ATR–Syntax–With–Some–Semantics tagset". The point about numbers, prices, times, etc., is that in many kinds of document, they are devilishly *frequent*. Hence one could conceive of uses for a tagger which accurately assigns this class of tag, within applications such as document scanning and information retrieval, among other places.

Verbal vs. Ordinary Adjectives And Nouns Arguably a problem with the tagsets which have been used so far in large–scale tagging experiments, has been the lack of an adequate treatment of *verbal* (as opposed to *ordinary*) *adjectives* and *nouns* (forms 1,5,9 of Table 6; contrast forms 2,6,10).[21] CLAWS conflates forms 1 and 2; 5 and 6; and 9 and 10. UPenn conflates 5 and 6. It assigns two different tags to 1 and 2, and to 9 and 10; however, the tag chosen for 1 is also the tag for 3,4,7,8; and the tag chosen for 9 is also the tag for 11 and 12. In contrast, the ATR tagsets feature different tags for cases 1 and 2; 5 and 6; and 9 and 10; the tag for case 1 differs from all of tags 2–12; and the tag for case 9 differs from all other cases 1–12.

Thus ATR can, but the other tagsets cannot, distinguish between "of a retiring_JJVVG employee" and "of a retiring_JJ nature"; between "a forced_JJVVN march" and "a forced_JJ smile";[22] and between "Hog calling_NVVG is a dying art" and "Bill has found his calling_NN1". Further, both senses of Chomsky's sentence "Flying planes can be dangerous" receive the same tagging by UPenn, but not by ATR (nor by CLAWS).[23]

Why does this matter? One place it matters is in machine translation. It makes sense that cases 1 and 9 should be translated differently from cases 2 and 10, since the former can be thought of as reflecting "regular lexical processes", whereas the latter are the result of "lexicalization", hence highly idiosyncratic. It would be absurd, in a process as complex as translation, to claim

[20] American slang for 25 cents

[21] Terminology of (Kruisinga, 1931). (Long, 1961) uses *participial adjectives* and *gerundial nouns* vs. *adjectives*, *nouns*.

[22] Again, the UPenn tagset does make these two distinctions, but then throws away their utility by using for case 1 the same tag as for cases 3,4,7,8, and for case 9 the same tags as for cases 11, 12, causing much potential confusion to a parser, e.g. with the Chomskian chestnut quoted below.

[23] Further, UPenn tags identically all *three* senses of e.g. "Singing lessons can be fun." In practice, the UPenn WSJ Treebank apparently fails to consistently capture any patterns over -ed and -ing adjectives and nouns. There is only mild correspondance between the tagging decisions prescribed for these forms in the UPenn Tagging Guidelines (1995 edition; contact sparnum@unagi.cis.upenn.edu), and those actually made in the WSJ Treebank. What results is relatively patternless labelling. For instance, in a 14,900–word sample of latest–version (0.75) WSJ Treebank, taken from three widely separated places in the corpus, only 93 of 159 -ed and -ing adjectives and nouns (58%) were correctly labelled with respect to the Tagging Guidelines.

-ing/-ed Form	ATR	CLAWS	UPENN
(1) The *sleeping* baby	JJVVG	JJ	VBG
(2) An *interesting* idea	JJ	JJ	JJ
(3) Ed is *running* away	VVG	VVG	VBG
(4) The man *running* away	VVG	VVG	VBG
(5) A *sleeping* pill	NVVG	NN1	NN
(6) He makes a good *living*	NN1	NN1	NN
(7) *Finding* gold is hard	VVG	VVG	VBG
(8) *Speaking* softly helps	VVG	VVG	VBG
(9) The *offered* amendment	JJVVN	JJ	VBN
(10) A *forced* smile	JJ	JJ	JJ
(11) Ed has *sold* his farm	VVN	VVN	VBN
(12) The man *given* $5	VVN	VVN	VBN

Table 7: Tagging Verbal Adjectives and Nouns With ATR, CLAWS and UPenn Tagsets

that "form x in English is translated via form y in French". But what we do find is a tendency to translate the two forms using a different gamut of structures.

Informant work in French, Japanese, and Korean suggests a tendency to translate case–1 forms (JJVVGs) via participles, and case–2 forms (JJs ending in –ing, often "lexicalized") with adjectives. Further, one author conducted an informal test using the 1986 Canadian Hansard French/English database,[24] in which 10 JJVVGs and 10 JJs ending in -ing were selected at random from the ATR Treebank. The first "adjectival" occurrence of each of these words in the 1986 Hansards was located, along with its French translation. The structural types of the translations were noted and tabulated. The "translation profile" which emerged of the -ing–form JJs was very different from that of the JJVVGs. Whereas in 4 cases, the JJs were translated via unambiguous adjectives, this never occurred for the JJVVGs. In both cases, 3 words were translated via present participles (-ing forms); but other than that, the entire profile was totally different for the two cases.

4.2.2 Confront The Unknown–Word And –Tag Problems

We know of no attempts to date to quantify the unknown–word and unknown–tag problems (viz., the word to be tagged (a) has never been encountered in the training corpus (unknown–word); or (b) is in the training corpus, but not with the tag which it needs to be assigned in the case at hand (unknown–tag).)

Table 7 shows the findings of a detailed exploration of the unknown–word problem involving the ATR Treebank and the UPenn WSJ Treebank. It just happens that the UPenn WSJ and ATR vocabularies each have 75% coverage of the other. (I.e. 75% of the different words (*types*) occurring in ATR figure on the list of types occurring in the UPenn WSJ Treebank.) We took great care to make the comparison as meaningful as possible, by (a) mapping all words to lowercase before comparing the two wordlists; (b) omitting consideration of plain numbers and digit sequences, digit–based words except meaningful ones like 9–foot, "non–words" of many stripes (e.g. black@itl.atr.co.jp, hellooooooo); and (c) compensating for any "tokenization" differences between

[24] supplied by the Linguistic Data Consortium (sparnum@unagi.cis.upenn.edu)

Covering Database	Covered Database	Category of Coverage	Coverage
UPenn WSJ Treebank	ATR Treebank	wordlist	75%
ATR Treebank	UPenn WSJ Treebank		75%
UPenn WSJ Treebank	ATR Treebank	running words	94%
ATR Treebank	UPenn WSJ Treebank		94%
UPenn WSJ Treebank	ATR Treebank	sentences	69%
CUVOALD92 Dictionary	ATR Treebank		60%
CUVOALD92 Dictionary + UPenn WSJ Treebank	ATR Treebank		80%

Table 8: Mutual Coverage Statistics For ATR and UPenn Treebanks

the two treebanks (e.g. UPenn converts "$500" into "$ 500", while ATR leaves it as is). Still, for various reasons, we can only guarantee the first two figures cited to within 5%.

We were even more careful in calculating the coverage for running words. I.e. what percent of all the word occurrences (*tokens*) in the UPenn WSJ Treebank (over 1 million) figure on the list of types in the ATR Treebank? And vice–versa. Here our answer, 94%, is estimated to within 1%. And here again, it just happened that the same answer applied in both directions. Thus, if one selects a word at random from, say, the UPenn WSJ Treebank, the chances are 94 in 100 that it is in the list of words occurring in the ATR Treebank.

So far, the unknown–word problem may appear fairly harmless. However, a further finding remains. We calculated the distribution of unknown words among sentences in the ATR Treebank. I.e. we calculated the percentage of ATR–Treebank sentences within which one or more words are unknown to the UPenn WSJ Treebank. (To ensure that the non–covered words were "real words", we also removed from consideration in this test all last names and names of cities. This represents a decision that e.g. "Martin" and "Nevada" are "words", whereas, say, "Hogsbristle" and "Oshkosh" are not.) That percentage, estimated to within 1%, was 69%! That is, about 3 of every 10 ATR–Treebank sentences are not "covered" by the UPenn WSJ Treebank! This suggests that in real–world tagging, the unknown–word problem is a serious one.

Further, we tested the coverage provided by a "dictionary", in a more conventional sense of the term than the one often used in tagging research.[25] That is, we tested the sentence–wise coverage of the ATR Treebank, by the CUVOALD92 Dictionary,[26] an expanded, computer–usable version, containing inflected forms, etc., of the Oxford Advanced Learner's Dictionary Of Current English[27] Again we omitted all last and city names, and again we verified carefully that only "real words" were counted in the comparison process. Results were that 60% of ATR–Treebank sentences were covered by CUVOALD92. Finally, even when we used both the UPenn–WSJ Treebank and the CUVOALD92 Dictionary, coverage of ATR–Treebank sentences was still only 80%. One in five sentences is not covered using this "dictionary".

We have made a start on a similar analysis of the unknown–tag problem; our results are shown in Figure 1. Crucially, we do not yet have figures on the distribution of unknown tags over (ATR– and UPenn–Treebank) sentences. However, one can see the effect of increasing tagset size on the simple incidence of unknown tags. For the ATR–Full tagset, unknown tags represent 8% of running

[25] viz., a list of all words in some tagged corpus, and the tags with which each word is associated once or more

[26] produced by Roger Mitton; available from: ftp://black.ox.ac.uk/ota/dicts/710

[27] Third Edition, Oxford University Press, 1974.

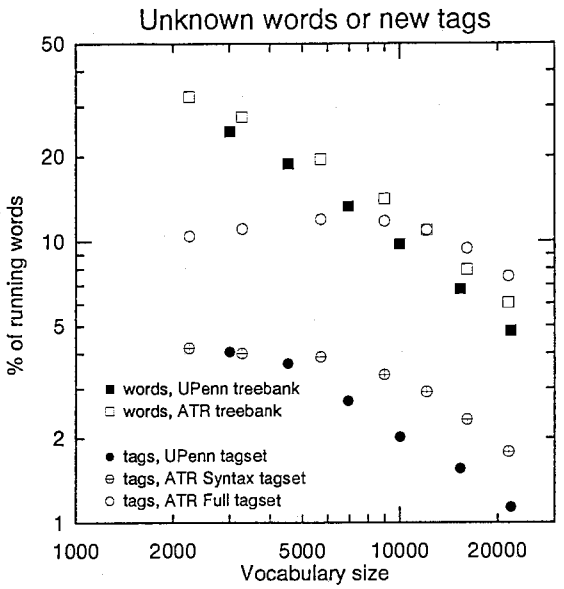

Figure 4: Percentage of running words in ATR and UPenn test corpora unknown in or having tags not used in the training set, as a function of training–set vocabulary size. Words consisting entirely of digits or punctuation are ignored. ATR training set, thus purged, contains 271,852 running words and a vocabulary of 21,627; UPenn, 885010 and 35,756, respectively.

words; for the UPenn tagset, around 1%.

4.3 The Non–Dictionary

We have attempted to tag using a more–detailed tagset, on a comprehensive treebank, and to confront the unknown–word and unknown–tag issues. What tools did we use, and how far did we get?

We call our approach the Non–Dictionary, or dictionaryless tagger. Why throw away the dictionary? Given the magnitude of the unknown–word and unknown–tag problems, well–developed means are necessary *anyway* of dealing with these cases of dictionary failure. More generally, the wider–ranging the treebank being tagged, and the larger and more detailed the tagset employed, the more quixotic it is to think that the universe of tags can be listed for a given word: "pumpkin" becomes an adjective when it is listed as the color of a sweater in the L.L. Bean catalog; "The" and "An" turn out to be first names in a text discussing the teaching of English As A Second Language in Southeast Asia; "As" shows up as a plural proper noun, on the sports page, as the name of a baseball team. It does not follow that the dictionary is a hindrance; but by pushing a dictionaryless approach as far as possible, we can concentrate on unknown–word and –tag issues, and later factor in a dictionary if we wish.

So, we in effect consider every word for tagging to be an unknown word. Instead of asking which tags have been seen for the word being tagged—in our training set, in an online dictionary, or in either place—we ask about: parts of words (sometimes formal affixes, sometimes not); certain "whole words"; the words surrounding the word being tagged; characteristics of the overall sentence; tags (or features of tags) which the tagger has already assigned; etc. We attempt to capture "trends" in a tagged treebank, trends which have to do with groups of words but which are much more varied and subtle than the tendency of specific part-of-speech trigrams to occur, or of a given word to have

been tagged a certain way a certain percent of the time.[28] (Brill, 1994; Black et al., 1992) exploit somewhat similar trends, but, in the first case, a different modeling approach is used, and in the second case, while a similar model to ours is used, crucially, (a) a dictionary is employed, (b) only self–organized questions are asked of the data (see below), and (c) a simpler tagset (mapped–down CLAWS) is employed.

So far the questions we have utilized are mainly aimed at doing syntactic tagging. We are at work, however, on many additional questions for use in syntactic–plus–semantic tagging. We generate questions both by hand and via self–organized methods, and we apply these questions to our training data by means of statistical decision trees. The outcome of the tagging process is essentially a probability distribution for each tag sequence for a sentence, over all tags in the tagset.

4.4 The Model

4.4.1 Mutual Information (MI) Bits

In addition to asking about affixes, capitalization, etc. of words in isolation, we can ask whether a given word is a member of a particular class of words.[29] We define word classes using the self–organizing approach of (Brown et al., 1992)—automatic clustering on large, untagged corpora, in this case 20,000,000 words of Wall–Street–Journal text. We assign each of the 70,000 most frequent words in this database to its own class, then iteratively merge the two classes which are most often used in similar situations. Specifically, if c_i represents the ith class, the mutual information of class bigram pairs is:

$$I \equiv \sum_{c_1,c_2} p(c_1,c_2) \log \frac{p(c_1,c_2)}{p(c_1)p(c_2)}. \tag{1}$$

We find the pair of classes whose merger into a single class will least decrease the mutual information (Ushioda, 1996). By keeping track of the order in which classes are merged, we can define a binary tree which spans all levels of detail from one class per word to a single class for all words. When these classes are utilized for constructing a decision–tree tagging model (see below), the decision tree can determine what level of detail to exploit.

4.4.2 Decision Trees

Decision trees are a formalization of the game of "20 questions" (Breiman et al., 1984; Black et al., 1992). The model consists of a tree–structured set of questions, with a probability distribution associated with each leaf of the tree. To estimate a conditional distribution using the tree, follow a path from the root to a leaf based on answers to the questions at each node. The leaf's associated distribution is the estimator. Training a decision tree model requires two steps: first, picking a question to ask at each node; and second, determining a probability distribution for each leaf, using the distribution of events in the training set which reach each node. As discussed in (Black et al., 1992), at each node we choose from among all possible questions (that is, all possible bits describing the current word and its context) that question which maximizes entropy reduction.

Assigning a tag is a two–stage process. First, a decision tree assigns one of 20 "generalized parts–of–speech"[30] (*GPOS's*) to the word based on a large set of word(–part) and context questions.

[28] as is relied upon in e.g. (Merialdo, 1994)

[29] In asking both manually–created and self–organized questions, we follow (Magerman, 1994).

[30] actually a value for the feature "pos" for the tag, as our tagset is feature–based

Second, a separate decision tree assigns a tag to the word based on an additional large set of word(–part) and context questions as well as its predicted GPOS. In this second stage, there is a separate decision tree for each GPOS. Breaking the process up this way allows us to concentrate on different word characteristics and different aspects of the context for different classes of tag.

4.4.3 The Tagging Process

Tagging proceeds from left to right, with the goal of maximizing the joint probability of the tag sequence for the entire sentence. That is, we find the set of tags $\{\hat{t}_1, \hat{t}_2, ..., \hat{t}_N\}$, where $\hat{t}_i$ is the predicted tag for the ith word of the N word sentence $w_1\ w_2\ ...\ w_N$, which maximizes

$$P \equiv p(\hat{t}_1, \hat{t}_2, ...\hat{t}_N | w_1, w_2, ...w_N) \tag{2}$$

$$= \prod_{i=1}^{N} p(\hat{t}_i | w_1, ..., w_N, \hat{t}_1, ..., \hat{t}_{i-1}) \tag{3}$$

Decision trees are used to extract relevant features from the conditions in these distributions. Note that we have not invoked the Markov assumption here—the predicted tag for even the last word of the sentence can, in principle, depend on the first word and its predicted tag. Whether this dependence in fact shows up in our models depends on whether the decision trees find it to be important for the training set. If we represent the deterministic process of using the answers to context-dependent questions to find a leaf in the tree as:

$$L_i \equiv leaf\ to\ which\ the\ context\ w_1, ..., w_N, \hat{t}_1, ..., \hat{t}_{i-1}\ leads, \tag{4}$$

and the probability distribution associated with leaf L as $\hat{p}_L$, then the decision trees approximate the required conditional distributions by

$$p(\hat{t}_i | w_1, ..., w_N, \hat{t}_1, ..., \hat{t}_{i-1}) \approx \hat{p}_{L_i}(\hat{t}_i) \tag{5}$$

and the function our search procedure tries to maximize is,

$$\prod_{i=1}^{N} \hat{p}_{L_i}(\hat{t}_i) \approx P \tag{6}$$

One final technicality is that we split the tagging process into two parts, first assigning a GPOS using one decision tree, then a tag using a separate decision tree specialized for the predicted GPOS. Thus in practice, the GPOS prediction uses the conditions above, while tag prediction uses these conditions plus the GPOS predicted for the current word.

When the first word of a sentence is considered, the context consists of the words and their arrangement in the sentence. The decision tree predicts the probability of each GPOS for this word in this context. Next, for each predicted "generalized part–of–speech", the appropriate decision tree is used to evaluate the probability of each possible tag. A search over this space determines the overall ranking for each tag. Then, the next word is considered. Relevant questions now include both the tag-independent questions used for the first word of the sentence, and questions which depend on the tag of the first word. For each different tag assigned to the first word, a set of GPOS's and then a set of tags are predicted. A search over the space of first–and–second–word tag–pairs determines overall ranking. This procedure continues until every word in the sentence has been tagged.

Our overall choice of the "best" tag for each word is intended to maximize the joint probability of the entire set of tags. This means we must evaluate the probability for a set of tag sequences which grows exponentially with the length of the sentence. We can either exhaustively enumerate and score all the cases (which is reasonable for a small tagset such as UPenn), or use a stack decoder algorithm (Bahl et al., 1983; Jelinek, 1969; Paul, 1990) to search through the most probable candidates (as is necessary for the ATR–Full tagset).

4.4.4 Example Questions

Here is a sampling of decision–tree questions created by our team grammarian.

Context Questions: (1) For the word being tagged: (a) position within sentence; (b) quadrant of sentence; (2) Final word of sentence: (a) question mark; (b) period or exclamation point; (3) Anywhere in sentence: (a) by (b) than; (4) For word sequences including word being tagged: (a) Specific ditto–tag words; (b) Any of large list of likely contexts for particular tag or GPOS; (c) Any of list of likely contexts for particular word used in particular sense—this for many words which share a semantic identity.

Word Questions: (asked of all words within two positions of the word being tagged, plus the word itself): (1) How many letters long (2) Contains "at–sign" (for email addresses, etc.) (3) any kind of determiner, article, pronoun; (4) ends in probable adjective suffix, yet not on exception list; (5) adjective in –wide (complex set of conditions: either the word "wide"; or word ending in –wide, and having either a hyperbolic prefix, or a number in digits or words as a substring); (6) on list of words, signalling start of subject noun phrase (and not on exception list); (7) has "time–adverb" prefix; (8) contains hyphenated preposition as "midstring"; (9) on list of synonyms for "remember"; (10) contains name of wild animal.

4.5 Experimental Results

The focus of the research being reported here is tagging with the ATR tagsets, on the ATR/Lancaster Treebank. As a point of reference for our results, however, we have also tagged the one publicly–available corpus, the UPenn Wall-Street–Journal Treebank, for which there are results utilizing various tagging approaches.

UPenn training and testing sets used consist of random sentences from the UPenn WSJ Treebank[31] (1,072,755 words of training, 133,293 smoothing, 49,624 testing data).[32] The random–document sets consist of randomly–selected documents from the ATR Treebank (319,903 words of training, 38,667 smoothing, 60,667 testing data). ATR random–sentence sets consist of randomly–selected sentences from the ATR Treebank (388,058 words of training, 43,189 smoothing, 12,150 testing data). Clearly, the random–document sets represent a fairer approximation of real–world tagging tasks.

Obviously, it is harder to tag with the ATR–Full tagset than with the UPenn tagset, but how much harder? We have tried to quantify the inherent difficulty of the various tasks for comparison. Table 9 displays the results of a "trivial tagger", which uses the most frequently seen tag for each known word, and the most frequent overall tag for every unknown word.[33] This provides a convenient baseline for judging the difficulty of the tagging tasks. The first column of Table 9 gives

[31] Version 0.75; annotated by the Penn Treebank Project; copyright University of Pennsylvania

[32] This includes every token that receives a tag. Approximately 200,000 of these are punctuation tags.

[33] We considered using a Hidden Markov Model for these comparisons, but felt it would not be informative because of the complexity of the task.

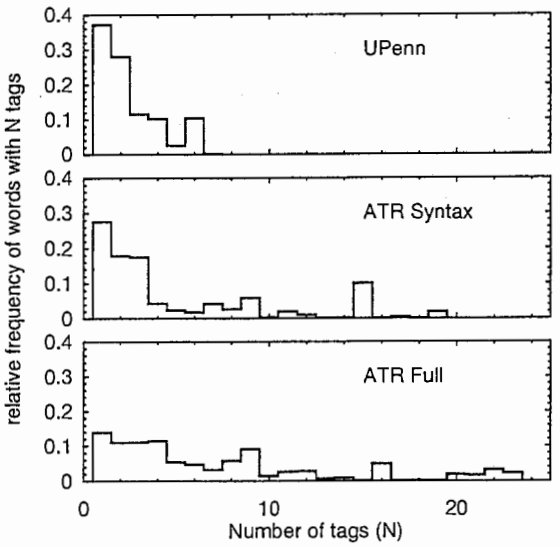

Figure 5: Histograms showing relative frequency of occurrence of words as function of number of distinct tags with which word is associated. Not shown: >25 (ATR–Full).

tagset	corpus	trivial % correct	perplexity training set	perplexity testing set
UPenn	UPenn	89.6	1.18	1.16
ATR Syntax	sentence	82.4	1.30	1.19
	document	83.6	1.30	1.26
ATR Full	sentence	69.3	1.73	1.36
	document	69.3	1.72	1.57

Table 9: "Trivial tagger": results

the tagging accuracy for this trivial tagger; the second and third show the perplexity of the training and testing data with respect to this model. We interpret the difference between perplexities for the training and testing sets to mean that we are still in a data–limited regime. In other words, the estimates differ in the case of ATR–Syntax and ATR–Full because the sample sizes are not large enough to provide stable estimates, whereas for UPenn they are. As a final means of comparing tagging difficulty among the three tagsets, we display Figure 5, which shows relative frequency of words with N tags, for each of the tagsets. The largest number of tags for a single word in the UPenn training set is 7, accounting for 0.1% of the running words. By contrast, 21.9% of the running words in the ATR–Syntax training set and 33.5% of the running words in the ATR–Full training set have more than 7 tags. The maximum for ATR–Syntax is 19 tags (1.9% of running words; 8.3% for ATR–Full).

Results are shown in Table 10, in several categories:[34] For "% correct" the set is every word in the test set; for KWT, only known words—those words which also appeared in the training set;

[34](EB 1999): Again, as noted at the beginning of this section, these results are out of date. Our current figures for ATR Full, document test set, is 85% on our "golden standard" test set (see Section 3).

tagset	corpus	% correct	KWT	KWKT	KWUT	UW
UPenn	UPenn	96.0	96.7	99.6	61.0	91.9
ATR	sentence	92.6	94.7	95.2	52.2	82.9
Syntax	document	90.8	93.8	94.6	41.2	79.6
ATR	sentence	76.5	79.4	83.6	8.5	63.7
Full	document	71.8	76.8	81.7	8.2	53.9

Table 10: Non–Dictionary tagger: results

tagset	full model	w/o MI bits	only MI bits
UPenn	96.0	95.3	94.6
ATR Syntax	90.8	89.9	86.1
ATR Full	71.8	68.8	69.4

Table 11: Percentage of running words tagged correctly for models which ignore mutual information bits or which use *only* mutual information bits. Results using both mutual–information and human–created questions shown for comparison. The ATR results are for the document-random test set.

for KWKT, only known words with known tags; for KWUT, only known words with an unknown tag; for UW, only unknown words. The results indicate that our methods work reasonably well on unknown words, and unknown tags for known words, although not on unknown tags for the ATR–Full tagset. To date, our efforts have largely concentrated on the ATR syntax tagset; we expect that work on questions suitable for the semantic parts of the ATR tagset will improve performance there.

All the results shown here use the mutual information bits described in 4.4.1. As shown in Table 11, we have found that incorporating these bits yields a statistically significant improvement, even though the vocabulary they use is specific to the WSJ corpus.[35]

Our plans for further research include exploring methods of factoring "dictionary" information (i.e. tag distribution by word in training data) into our models; manual question–creation for ATR–Full, while improving ATR–Syntax questions; and possibly clustering a much larger dataset for improved MI questions.[36]

4.6 Experiments In Tagging Improvement (1)

Note: One of the directions we have pursued in our efforts to improve prediction of tag assignment in English text, is the use of information outside the sentence in which the word occurs which is

[35]The WSJ data from which our MI bits were created included the million words corresponding to the UPenn WSJ Treebank. Hence the 0.7% contribution of the MI bits to UPenn Treebank tagging results should be interpreted cautiously. However, the performance of these bits on ATR–Treebank tasks, e.g. the 0.9% contribution to our ATR–Syntax score, suggests that most or all of the 0.7% contribution to the UPenn score would stand if we reclustered omitting these million words from the 20–million–word dataset used.

[36](EB 1999): All of these means of improving results are currently being pursued, with the exception of further MI bit clustering.

being tagged. This subsection presents the first of two sets of experiments undertaken towards that end. The upshot of the two sets of experiments, both presented in this report, has been to encourage us in the direction of working extrasentential information into our routines for tag prediction, and based on the results of this first set of experiments, in parse prediction as well. The specific means we ultimately choose of incorporating such factors into our predictive software are currently being determined. In fine, expanding the sources of information which are interrogated in the effort to predict tag assignments is one theme of the work we will be pursuing in the successor laboratory to ITL, in order to fulfill our goal for the new research period, of realizing the potential of our linguistic analysis approach, by bringing prediction of tag and parse assignments up to near-human levels of accuracy. The original presentation of our experimental work follows immediately below:

If a person or device wished to predict which words or grammatical constructions were about to occur in some document, intuitively one of the most helpful things to know would seem to be which words and constructions occurred within the last half–dozen or dozen sentences of the document. Other things being equal, a text that has so far been larded with, say, mountaineering terms, is a good bet to continue featuring them. An author with the habit of ending sentences with adverbial clauses of confirmation, e.g. "as we all know", will probably keep up that habit as the discourse progresses.

Within the field of language modelling for speech recognition, maintaining a cache of words that have occurred so far within a document, and using this information to alter probabilities of occurrence of particular choices for the word being predicted, has proved a winning strategy (Kuhn et al., 1990). Models using *trigger pairs* of words, i.e. pairs consisting of a "triggering" word which has already occurred in the document being processed, plus a specific "triggered" word whose probability of occurrence as the next word of the document needs to be estimated, have yielded perplexity[37] reductions of 29–38% over the baseline trigram model, for a 5–million–word Wall Street Journal training corpus (Rosenfeld, 1996).

This subsection introduces the idea of using trigger–pair techniques to assist in the prediction of rule and tag occurrences, within the context of natural–language parsing and tagging. Given the task of predicting the correct rule to associate with a parse-tree node, or the correct tag to associate with a word of text, and assuming a particular class of parsing or tagging model, we quantify the information gain realized by taking account of rule or tag trigger-pair predictors, i.e. pairs consisting of a "triggering" rule or tag which has already occurred in the document being processed, plus a specific "triggered" rule or tag whose probability of occurrence within the current sentence we wish to estimate.

In what follows, subsection 4.7 provides a basic overview of trigger–pair models. subsection 4.8 describes the experiments we have performed, which to a large extent parallel successful modelling experiments within the field of language modelling for speech recognition. In the first experiment, we investigate the use of trigger pairs to predict both rules and tags over our full corpus of around a million words. The subsequent experiments investigate the additional information gains accruing from trigger–pair modelling when we know what sort of document is being parsed or tagged. We present our experimental results in subsection 4.9, and discuss them in subsection 4.10. In subsection 4.11, we present some example trigger pairs; and we conclude, with a glance at projected future research, in subsection 4.12.

[37] See Section 2.

4.7 Background

Trigger–pair modelling research has been pursued within the field of language modelling for speech recognition over the last decade (Beeferman et al., 1997; Della Pietra et al., 1992; Kupiec, 1989; Lau, 1994; Lau et al., 1993; Rosenfeld, 1996).

Fundamentally, the idea is a simple one: if you have recently seen a word in a document, then it is more likely to occur again, or, more generally, the prior occurrence of a word in a document affects the probability of occurrence of itself and other words.

More formally, from an information–theoretic viewpoint, we can interpret the process as the relationship between two dependent random variables. Let the outcome (from the alphabet of outcomes $\mathcal{A}_Y$) of a random variable Y be observed and used to predict a random variable X (with alphabet $\mathcal{A}_X$). The probability distribution of X, in our case, is dependent on the outcome of Y.

The average amount of information necessary to specify an outcome of X (measured in bits) is called its *entropy* $H(X)$ and can also be viewed as a measure of the average ambiguity of its outcome:[38]

$$H(X) = \sum_{x \in \mathcal{A}_X} -P(x) \log_2 P(x) \tag{7}$$

The *mutual information* between X and Y is a measure of entropy (ambiguity) reduction of X from the observation of the outcome of Y. This is the entropy of X minus its *a posteriori* entropy, having observed the outcome of Y.

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= \sum_{x \in \mathcal{A}_X} \sum_{y \in \mathcal{A}_Y} P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)} \end{aligned} \tag{8}$$

The dependency information between a word and its history may be captured by the *trigger pair*.[39] A trigger pair is an ordered pair of words t and w. Knowledge that the trigger word t has occurred within some *window* of words in the history, changes the probability estimate that word w will occur subsequently.

Selection of these triggers can be performed by calculating the average mutual information between word pairs over a training corpus. In this case, the alphabet $\mathcal{A}_X = \{w, \overline{w}\}$, the presence or absence of word w; similarly, $\mathcal{A}_Y = \{t, \overline{t}\}$, the presence or absence of the triggering word in the history.

This is a measure of the effect that the knowledge of the occurrence of the triggering word t has on the occurence of word w, in terms of the entropy (and therefore perplexity) reduction it will provide. In all our experiments, the first term of equation (3) makes by far the largest contribution. Clearly, in the absence of other context (i.e. in the case of the *a priori* distribition of X), this information will be additional. However, once related contextual information is included (for example by building a trigram model, or, using other triggers for the same word), this is no longer strictly true.

Once the trigger pairs are chosen, they may be used to form constraint functions to be used in a maximum–entropy model, alongside other constraints. Models of this form are extremely versatile,

[38] A more intuitive view of entropy is provided through *perplexity* (Jelinek et al., 1977) which is a measure of the number of choices, on average, there are for a random variable. It is defined to be: $2^{H(X)}$.

[39] For a thorough description of trigger-based modelling, see (Rosenfeld, 1996).

allowing the combination of short– and long–range information. To construct such a model, one transforms the trigger pairs into *constraint functions* $f(t, w)$:

$$f(t,w) = \begin{cases} 1 & \text{if } t \in \text{history and} \\ & \text{next word} = w \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The expected values of these functions are then used to constrain the model, usually in combination of with other constraints such as similar functions embodying uni–, bi– and trigram probability estimates.

(Beeferman et al., 1997) models more accurately the effect of distance between triggering and triggered word, showing that for non–self–triggers,[40] the triggering effect decays exponentially with distance. For self–triggers,[41] the effect is the same except that the triggering effect is lessened within a short range of the word. Using a model of these distance effects, they are able to improve the performance of a trigger model.

We are unaware of any work on the use of trigger pairs in parsing or tagging. In fact, we have not found any previous research in which extrasentential data of any sort are applied to the problem of parsing or tagging.

4.8 The Experiments

4.8.1 Experimental Design

In order to investigate the utility of using long–range trigger information in tagging and parsing tasks, we adopt the simple mutual–information approach used in (Rosenfeld, 1996). We carry over into the domain of tags and rules an experiment from Rosenfeld's paper the details of which we outline below.

The idea is to measure the information contributed (in bits, or, equivalently in terms of perplexity reduction) by using the triggers. Using this technique requires special care to ensure that information "added" by the triggers is indeed additional information.

For this reason, in all our experiments we use the unigram model as our base model and we allow only one trigger for each tag (or rule) token.[42] We derive these unigram probabilities from the training corpus and then calculate the total mutual information gained by using the trigger pairs, again with respect to the training corpus.

When using trigger pairs, one usually restricts the trigger to occur within a certain window defined by its distance to the triggered token. In our experiments, the window starts at the sentence prior to that containing the token and extends back W (the window size) sentences. The choice to use sentences as the unit of distance is motivated by our intention to incorporate triggers of this form into a probabilistic treebank–based parser and tagger, such as (Black et al., 1998; Black et al., 1997; Brill, 1993; Brill, 1994; Collins, 1996; Jelinek et al., 1994; Magerman, 1995; Marquez et al., 1997; Ratnaparkhi, 1997). All such parsers and taggers of which we are aware use only intrasentential information in predicting parses or tags, and we wish to remove this information,

[40] i.e. words which trigger words other than themselves

[41] i.e. words which trigger themselves

[42] By rule assignment, we mean the task of assigning a rule–name to a node in a parse tree, given that the constituent boundaries have already been defined.

1868 documents
80299 sentences
904431 words (tag instances)
1622664 constituents (rule instances)
1873 tags utilized
907 rules utilized
11.3 words per sentence, on average

Table 12: Characteristics of Training Set (Subset of ATR/Lancaster General–English Treebank)

as far as possible, from our results [43]. The window was not allowed to cross a document boundary. The perplexity of the task before taking the trigger–pair information into account for tags was 224.0 and for rules was 57.0.

The characteristics of the training corpus we employ are given in Table 12. The corpus, a subset[44] of the ATR/Lancaster General–English Treebank (Black et al., 1996), consists of a sequence of sentences which have been tagged and parsed by human experts in terms of the ATR English Grammar, a broad–coverage grammar of English with a high level of analytic detail (Black et al., 1996; Black et al., 1997). For instance, the tagset is both semantic and syntactic, and includes around 2000 different tags, which classify nouns, verbs, adjectives and adverbs via over 100 semantic categories. As examples of the level of syntactic detail, exhaustive syntactic and semantic analysis is performed on all nominal compounds; and the full range of attachment sites is available within the Grammar for sentential and phrasal modifiers, and are used precisely in the Treebank. The Treebank actually consists of a set of documents, from a variety of sources. Crucially for our experiments (see below), the idea[45] informing the selection of (the roughly 2000) documents for inclusion in the Treebank was to pack into it the maximum degree of document variation along many different scales—document length, subject area, style, point of view, etc.—but without establishing a single, predetermined classification of the included documents.[46]

In the first experiment, we examine the effectiveness of using trigger pairs over the entire training corpus. At the same time we investigate the effect of varying the window size. In additional experiments, we observe the effect of partitioning our training dataset into a few relatively homogeneous subsets, on the hypothesis that this will decrease perplexity. It seems reasonable that in different text varieties, different sets of trigger pairs will be useful, and that tokens which do not have effective triggers within one text variety may have them in another.[47]

To investigate the utility of partitioning the dataset, we construct a separate set of trigger pairs for each class. These triggers are only active for their respective class and are independent of each other. Their total mutual information is compared to that derived in exactly the same way from a random partition of our corpus into the same number of classes, each comprised of the same

[43] This is not completely possible, since correlations, even if slight, will exist between intra– and extrasentential information

[44] specifically, a roughly–900,000–word subset of the full ATR/Lancaster General–English Treebank (about 1.05 million words), from which all 150,000 words were excluded that were treebanked by the two least accurate ATR/Lancaster treebankers (expected hand–parsing error rate 32%, versus less than 10% overall for the three remaining treebankers)

[45] see (Black et al., 1996)

[46] as was done, say, in the Brown Corpus (Kucera and Francis, 1967)

[47] Related work in topic–specific trigram modelling (Lau, 1994) has led to a reduction in perplexity.

Part. 1: *Source*		Part. 4: *Source Plus Document Type*		Part. 5: *Source Plus Domai*	
Class Name	Sents	Class Name	Sents	Class Name	Sen
1: Assoc. Press, WSJ	8851	1: Legislative (incl. *Src.*2)	5626	1: Recreation	354
2: Canadian Hansards	5002	2: Transcripts (incl. *Src.*4)	44287	2: Business	205
3: General English	23105	3: News (incl. most *Src.*1)	8614	3: Science, Techn.	401
4: Travel–domain dialogs	43341	4: Polemical essays	5160	4: Humanities	222
Part. 2: *List Structure*		5: Reports; FAQs; listings	11440	5: Daily Living	89
Class Name	Sents	6: Idiom example sents	666	6: Health, Education	164
1: Contains lists	14147	7: Novels; stories; fables	741	7: Government, Polit.	176
2: Contains no lists	66152	8: Letters; diaries	1997	8: Travel	266
Part. 3: *Source Plus List Structure*		9: Legal cases; cnsttutns	1768	9: Social Sciences	361
Class Name	Sents			10: Idiom xmp. sents	66
1: Assoc. Press, WSJ	8851			11: Canad. Hansards	500
2: Canadian Hansards	5002			12: Asso. Press, WSJ	885
3: Contains lists (Gen.)	11998			13: Travel dialogs	43
4: Contains no lists (Gen.)	11117				
5: Travel–domain dialogues	43341				

Table 13: Training Set Partitions

number of documents.

Our training data partitions naturally into four subsets, shown in Table 13 as Partitioning 1 ("Source"). Partitioning 2, "List Structure", puts all documents which contain at least some HTML–like "List" markup (e.g. LI (=List Item))[48] in one subset, and all other documents in the other subset. By merging Partitionings 1 and 2 we obtain Partitioning 3, "Source Plus List Structure". Partitioning 4 is "Source Plus Document Type", and contains 9 subsets, e.g. "Letters; diaries" (subset 8) and "Novels; stories; fables" (subset 7). With 13 subsets, Partitioning 5, "Source Plus Domain", includes e.g. "Social Sciences" (subset 9) and Recreation (subset 1). Partitionings 4 and 5 were effected by actual inspection of each document, or at least of its title and/or summary, by one of the authors. The reason we included Source within most partitionings was to determine the extent to which information gains were additive.[49]

4.9 Experimental Results

4.9.1 Window Size

Figure 1 shows the effect of varying the window size from 1 to 500 for both rule and tag tokens. The optimal window size for tags was approximately 12 sentences (about 135 words) and for rules it was approximately 6 sentences (about 68 words). These values were used for all subsequent experiments. It is interesting to note that the curves are of similar shape for both rules and tags and that the optimal value is not the largest window size. Related effects for words are reported in (Lau, 1994; Beeferman et al., 1997). In the latter paper, an exponential model of distance is used

[48] All documents in our training set are marked up in HTML–like annotation.

[49] For instance, compare the results for Partitionings 1, 2, and 3 in this regard.

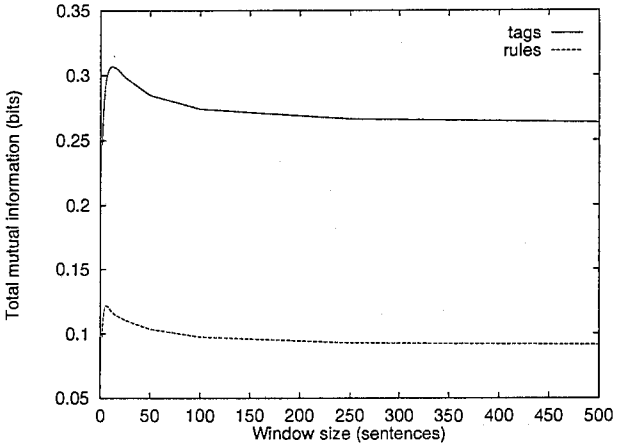

Figure 6: Mutual information gain varying window size

Partitioning	Perplexity reduction for tags		Perplexity reduction for rules	
	Meaningful partition	Random	Meaningful partition	Random
1: *Source*	28.40%	16.66%	15.44%	6.30%
2: *List Structure*	20.39%	18.71%	10.55%	7.46%
3: *Source Plus List Structure*	28.74%	17.12%	15.61%	6.50%
4: *Source Plus Document Type*	30.11%	18.15%	16.20%	6.82%
5: *Source Plus Domain*	31.55%	19.39%	16.60%	7.34%

Table 14: Perplexity reduction using class-specific triggers to predict tags and rules

to penalize large distances between triggering word and triggered word. The variable window used here can be seen as a simple alternative to this.

One explanation for this effect in our data is, in the case of tags, that topic changes occur in documents. In the case of rules, the effect would seem to indicate a short span of relatively intense stylistic carryover in text. For instance, it may be much more important, in predicting rules typical of list structure, to know that similar rules occurred a few sentences ago, than to know that they occurred dozens of sentences back in the document.

4.9.2 Class-Specific Triggers

Table 14 shows the improvement in perplexity over the base (unigram) tag and rule models for both the randomly–split and the hand–partitioned training sets. In every case, the meaningful split yielded significantly more information than the random split. (Of course, the results for randomly–split training sets are roughly the same as for the unpartitioned training set (Figure 6)).

4.10 Discussion

The main result of the work reported in this subsection is to show that analogous to the case of words in language modelling, a significant amount of extrasentential information can be extracted

#	Triggering Tag	Triggered Tag	I.e. Words Like These:	Trigger Words Like These:
1	NP1LOCNM	NP1STATENM	Hill, County, Bay, Lake	Utah, Maine, Alaska
2	JJSYSTEM	NP1ORG	national, federal, political	Party, Council, Department
3	VVDINCHOATIVE	VVDPROCESSIVE	caused, died, made, failed	began, happened, became
4	IIDESPITE	CFYET	despite	yet (conjunction)
5	DD	PPHO2	any, some, certain	them
6	PN1PERSON	LEBUT22	everyone, one, anybody	(not) only, (not) just
7	...	MPRICE	..., ..., ...	$452,983,000, $10,000
8	IIATSTANDIN	MPHONE22	at (sent.–final, +/–":")	913-3434 (follows area cd.)
9	IIFROMSTANDIN	MZIP	from (sent.–final, +/–":")	22314-1698 (postal zipcd.)
10	NNUNUM	NN1MONEY	25%, 12", 9.4m3	profit, price, cost

Table 15: Selected Tag Trigger–Pairs, ATR/Lancaster General–English Treebank

#	A Construction Like This:	Triggers A Construction Like This:
1a	Interrupter Phrase –> * Or –	Sentence –> Interr. P+Phrasal Constit (Non–S)
1b	*Example:* *, –	*Example:* * DIG. AM/FM TUNER
2a	VP –> Verb+Interrupter Phrase+Obj/Compl	Interrupter Phrase –> ,+Interrupter+,
2b	*Example:* starring—surprise, surprise—men	*Example:* , according to participants ,
3a	Noun Phrase –> Simple Noun Phrase+Numerical	Numerical –> Numcl +PrepP with Numcl Obj
3b	*Example:* Lows around 50	*Example:* (Snow level) 6000 to 7000
4a	Verb Phrase –> Adverb Phrase+Verb Phrase	Auxiliary VP –> Model/Auxilliary Verb+Not
4b	*Example:* just need to understand it	*Example:* do not
5a	Question –> Be+NP+Object/Complement	Quoted Phrasal Constit –> "+Phrsl Constit+"
5b	*Example:* Is it possible?	*Example:* "Mutual funds are back."

Table 16: Selected Rule Trigger–Pairs, ATR/Lancaster General–English Treebank

#	Triggering Tag	Triggered Tag	I.e. Words Like These:	Trigger Words Like These:
1	VVNSEND	NP1STATENM	shipped, distributed	Utah, Maine, Alaska
2	NP1LOCNM	NP1STATENM	Hill, County, Bay, Lake	Utah, Maine, Alaska
For training–set document class Recreation (1) vs. for unpartitioned training set (2)				
3	VVOALTER	NN2SUBSTANCE	inhibit, affect, modify	tumors, drugs, agents
4	JJPHYS-ATT	NN2SUBSTANCE	fragile, brown, choppy	pines, apples, chemicals
For training–set document class Health And Education (3) vs. for unpartitioned training set (4)				
5	NN1TIME	NN2MONEY	period, future, decade	expenses, fees, taxes, prices
6	NP1POSTFRMNM	NN2MONEY	Inc., Associates, Co.	loans, damages, charges, prices
For training–set document class Business (5) vs. for unpartitioned training set (6)				
7	DD1	DDQ	this, that, another, each	which
8	DDQ	DDQ	which	which
For training–set document class Travel Dialogues (7) vs. for unpartitioned training set (8)				

Table 17: Selected Tag Trigger–Pairs, ATR/Lancaster General–English Treebank: Contrasting Trigger–Pairs Arising From Partitioned vs. Unpartitioned Training Sets

from the long–range history of a document, using trigger pairs for tags and rules. Although some redundancy of information is inevitable, we have taken care to exclude as much information as possible that is already available to (intrasentential–data–based, i.e. all known) parsers and taggers.

Quantitatively, the studies of (Rosenfeld, 1996) yielded a total mutual information gain of 0.38 bits, using Wall Street Journal data, with one trigger per word. In a parallel experiment, using the same technique, but on the ATR/Lancaster corpus, the total mutual information of the triggers for *tags* was 0.41 bits. This figure increases to 0.52 bits when tags further away than 135 tags (the approximate equivalent in words to the optimal window size in sentences) are excluded from the history. For the remainder of our experiments, we do not use as part of the history the tags/rules from the sentence containing the token to be predicted. This is motivated by our wish to exclude the intrasentential information which is already available to parsers and taggers.

In the case of tags, using the optimal window size, the gain was 0.31 bits, and for rules the information gain was 0.12 bits. Although these figures are not as large as for the case where intrasentential information is incorporated, they are sufficiently close to encourage us to exploit this information in our models.

For the case of words, the evidence shows that triggers derived in the same manner as the triggers in our experiments, can provide a substantial amount of new information when used in combination with sophisticated language models. For example, (Rosenfeld, 1996) used a maximum–entropy model trained on 5 million words, with only trigger, uni–, bi– and trigram constraints, to measure the test–set perpexity reduction with respect to a "compact" backoff trigram model, a well-respected model in the language–modelling field. When the top six triggers for each word were used, test–set perplexity was reduced by 25%. Furthermore, when a more sophisticated version of this model[50] was applied in conjunction with the SPHINX II speech recognition system (Huang et al., 1993), a 10-14% reduction in word error rate resulted (Rosenfeld, 1996). We see no reason why this effect should not carry over to tag and rule tokens, and are optimistic that long–range trigger

[50] trained on 38 million words, and also employing distance–2 N-gram constraints, a unigram cache and a conditional bigram cache (this model reduced perplexity over the baseline trigram model by 32%)

information can be used in both parsing and tagging to improve performance.

For words (Rosenfeld, 1996), *self–triggers*—words which triggered themselves—were the most frequent kind of triggers (68% of all word triggers were self–triggers). This is also the case for tags and rules. For tags, 76.8% were self-triggers, and for rules, 96.5% were self–triggers. As in the case of words, the set of self–triggers provides the most useful predictive information.

4.11 Some Examples

We will now explicate a few of the example trigger pairs in Tables 15–17. Table 15 Item 7, for instance, captures the common practice of using a sequence of points, e.g. ....., to separate each item of a (price) list and the price of that item. Items 8 and 9 are similar cases (e.g. "contact/call (someone) at:" + phone number; "available from:" + source, typically including address, hence zipcode). These correlations typically occur within listings, and, crucially for their usefulness as triggers, typically occur many at a time.

When triggers are drawn from a relatively homogeneous set of documents, correlations emerge which seem to reflect the character of the text type involved. So in Table 17 Item 5, the proverbial equation of time and money emerges as more central to Business and Commerce texts than the different but equally sensible linkup, within our overall training set, between business corporations and money.

Turning to rule triggers, Table 16 Item 1 is more or less a syntactic analog of the tag examples Table 15 Items 7–9, just discussed. What seems to be captured is that a particular style of listing things, e.g. * + listed item, characterizes a document as a whole (if it contains lists); further, listed items are not always of the same phrasal type, but are prone to vary syntactically. The same document that contains the list item "* DIG. AM/FM TUNER", for instance, which is based on a Noun Phrase, soon afterwards includes "* WEATHER PROOF" and "* ULTRA COMPACT", which are based on Adjective Phrases.

Finally, as in the case of the tag trigger examples of Table 7, text–type–particular correlations emerge when rule triggers are drawn from a relatively homogeneous set of documents. A trigger pair of constructions specific to Class 1 of the Source partitioning, which contains only Associated Press newswire and Wall Street Journal articles, is the following: A sentence containing both a quoted remark and an attribution of that remark to a particular source, triggers a sentence containing simply a quoted remark, without attribution. (E.g. *"The King was in trouble," Wall wrote.* triggers *"This increased the King's bitterness."*.) This correlation is essentially absent in other text types.

4.12 Conclusion

In this subsection, we have shown that, as in the case of words, there is a substantial amount of information outside the sentence which could be used to supplement tagging and parsing models. We have also shown that knowledge of the type of document being processed greatly increases the usefulness of triggers. If this information is known, or can be predicted accurately from the history of a given document being processed, then model interpolation techniques (Jelinek et al., 1980) could be employed, we anticipate, to exploit this to useful effect.

Future research will concentrate on incorporating trigger–pair information, and extrasentential information more generally, into more sophisticated models of parsing and tagging.[51] An obvious first extention to this work, for the case of tags, will be, following (Rosenfeld, 1996), to incorporate

[51](EB 1999): As stated in the opening note to this entire subsection, this work is ongoing.

#	Triggering Tag	Triggered Tag	I.e. Words Like:	Trigger Words Like:
1	NP1LOCNM	NP1STATENM	Hill, County, Bay	Utah, Maine, Alaska
2	JJSYSTEM	NP1ORG	national, federal	Party, Council
3	VVDINCHOATIVE	VVDPROCESSIVE	caused, died, made	began, happened
4	IIDESPITE	CFYET	despite	yet (conjunction)
5	DD	PPHO2	any, some, certain	them
6	PN1PERSON	LEBUT22	everyone, one	(not) only, (not) just
7	...	MPRICE	..., ..., ...	$452,983,000, $10,000
8	IIATSTANDIN	MPHONE22	at (sent.–final)	913-3434
9	IIFROMSTANDIN	MZIP	from (sent.–final)	22314-1698 (zip)
10	NNUNUM	NN1MONEY	25%, 12", 9.4m3	profit, price, cost

Table 18: Selected Tag Trigger–Pairs, ATR General–English Treebank

the triggers into a maximum–entropy model using trigger pairs in addition to unigram, bigram and trigram constraints. Later we intend to incorporate trigger information into a probabilistic English parser/tagger which is able to ask complex, detailed questions about the contents of a sentence. From the results presented here we are optimistic that the additional, extrasentential information provided by trigger pairs will benefit such parsing and tagging systems.

4.13 Experiments In Tagging Improvement (2)

Note (EB 1999): The work in the present subsection represents a followup to that of the previous subsection. Again, the upshot, for our work, is to encourage us to incorporate extrasentential inofrmation into our tag prediction, in one form or another, though not necessarily in any form directly linked to the work presented here. The original report of this work follows directly below:

It appears intuitively that information from earlier sentences in a document ought to help reduce uncertainty as to a word's correct part–of–speech tag. This is especially so for a large semantic and syntactic tagset such as the roughly–3000–tag ATR General English Tagset (Black et al., 1996; Black et al., 1998). And in fact, (Black et al., 1998) demonstrate a significant "tag trigger–pair" effect. That is, given that certain "triggering" tags have already occurred in a document, the probability of occurrence of specific "triggered" tags is raised significantly—with respect to the unigram tag probability model. Table 18, taken from (Black et al., 1998), provides examples of the tag trigger–pair effect.

Yet, it is one thing to show that extrasentential context yields a gain in information with respect to a unigram tag probability model. But it is another thing to demonstrate that extrasentential context supports an improvement in perplexity vis–a–vis a part–of–speech tagging model which employs state–of–the–art techniques: such as, for instance, the tagging model of a maximum entropy tag–n–gram–based tagger.

The present subsection undertakes just such a demonstration. Both the model underlying a standard tag-n-gram-based tagger, and the same model augmented with extrasentential contextual information, are trained on the 850,000–word ATR General English Treebank (Black et al., 1996), and then tested on the accompanying 53,000–word test treebank. Performance differences are measured, with the result that semantic information from previous sentences within a document is shown to help significantly in improving the perplexity of tagging with the indicated tagset.

In what follows, subsection 4.14 provides a basic overview of the tagging approach used (a maximum entropy tagging model employing constraints equivalent to those of the standard hidden Markov model). Section 4.15 discusses and offers examples of the sorts of extrasententially–based semantic constraints that were added to the basic tagging model. Section 4.16 describes the experiments we performed. Section 4.17 details our experimental results. Section 4.18 glances at projected future research, and concludes.

4.14 Tagging Model

4.14.1 ME Model

Our tagging model is a maximum entropy (ME) model of the following form:

$$p(t|h) = \gamma \prod_{k=0}^{K} \alpha_k^{f_k(h,t)} p_0 \tag{10}$$

where:

- t is tag we are predicting;
- h is the history (all prior words and tags) of t;
- γ is a normalization coefficient that ensures: $\Sigma_{t=0}^{L} \gamma \prod_{k=0}^{K} \alpha_k^{f_k(h,t)} p_0 = 1$;
- L is the number of tags in our tag set;
- α_k is the weight of trigger f_k;
- f_k are trigger functions and $f_k \epsilon \{0, 1\}$;
- p_0 is the default tagging model (in our case, the uniform distribution, since all of the information in the model is specified using ME constraints).

The model we use is similar to that of (Ratnaparkhi, 1996). Our baseline model shares the following features with this tagging model; we will call this set of features the basic n-gram tagger constraints:

1. $w = X \ \& \ t = T$
2. $t_{-1} = X \ \& \ t = T$
3. $t_{-2}t_{-1} = XY \ \& \ t = T$

where:

- w is word whose tag we are predicting;
- t is tag we are predicting;
- t_{-1} is tag to the left of tag t;
- t_{-2} is tag to the left of tag t_{-1};

Our baseline model differs from Ratnaparkhi's in that it does not use any information about the occurrence of words in the history or their properties (other than in constraint 1). Our model exploits the same kind of tag–n–gram information that forms the core of many successful tagging models, for example, (Kupiec, 1992), (Merialdo, 1994), (Ratnaparkhi, 1996). We refer to this type of tagger as a tag–n–gram tagger.

4.14.2 Trigger selection

We use mutual information (MI) to select the most useful trigger pairs (for more details, see (Rosenfeld, 1996)). That is, we use the following formula to gauge a feature's usefulness to the model:

$$\begin{aligned} MI(s,t) &= P(s,t)\log\frac{P(t|s)}{P(t)} \\ &+ P(s,\bar{t})\log\frac{P(\bar{t}|s)}{P(\bar{t})} \\ &+ P(\bar{s},t)\log\frac{P(t|\bar{s})}{P(t)} \\ &+ P(\bar{s},\bar{t})\log\frac{P(\bar{t}|\bar{s})}{P(\bar{t})} \end{aligned}$$

where:

- t is the tag we are predicting;

- s can be any kind of triggering feature.

For each of our trigger predictors, s is defined below:

Bigram and trigram triggers : s is the presence of a particular tag as the first tag in the bigram pair, or the presence of two particular tags (in a particular order) as the first two tags of a trigram triple. In this case, t is the presence of a particular tag in the final position in the n-gram.

Extrasentential tag triggers : s is the presence of a particular tag in the extrasentential history.

Question triggers : s is the boolean answer to a question.

This method has the advantage of finding good candidates quickly, and the disadvantage of ignoring any duplication of information in the features it selects. A more principled approach is to select features by actually adding them one-by-one into the ME model (Della Pietra et al., 1997); however, using this approach is very time-consuming and we decided on the MI approach for the sake of speed.

4.15 The Constraints

To understand what extrasentential semantic constraints were added to the base tagging model in the current experiments, one needs some familiarity with the ATR General English Tagset. For detailed presentations, see (Black et al., 1998; Black et al., 1996). An apercu can be gained, however, from Figure 7, which shows two sample sentences from the ATR Treebank (and originally from a Chinese take–out food flier), tagged with respect to the ATR General English Tagset. Each verb, noun, adjective and adverb in the ATR tagset includes a semantic label,

```
(_( Please_RRCONCESSIVE Mention_VVIVERBAL-ACT this_DD1 coupon_NN1DOCUMENT
when_CSWHEN ordering_VVGINTER-ACT

OR_CCOR ONE_MC1WORD FREE_JJMONEY FANTAIL_NN1ANIMAL SHRIMPS_NN1FOOD
```

Figure 7: Two ATR Treebank Sentences from Chinese Take-Out Food Flier (Tagged Only – i.e. Parses Not Displayed)

chosen from 42 noun/adjective/adverb categories and 29 verb/verbal categories, some overlap existing between these category sets. Proper nouns, plus certain adjectives and certain numerical expressions, are further categorized via an additional 35 "proper-noun" categories. These semantic categories are intended for any "Standard-American-English" text, in any domain. Sample categories include: "physical.attribute" (nouns/adjectives/adverbs), "alter" (verbs/verbals), "interpersonal.act" (nouns/adjectives/adverbs/verbs/verbals), "orgname" (proper nouns), and "zipcode" (numericals). They were developed by the ATR grammarian and then proven and refined via day-in-day-out tagging for six months at ATR by two human "treebankers", then via four months of tagset-testing-only work at Lancaster University (UK) by five treebankers, with daily interactions among treebankers, and between the treebankers and the ATR grammarian. The semantic categorization is, of course, in addition to an extensive syntactic classification, involving some 165 basic syntactic tags.

Starting with a basic tag-n-gram tagger trained to tag raw text with respect to the ATR General English Tagset, then, we added constraints defined in terms of "tag families". A tag family is the set of all tags sharing a given semantic category. For instance, the tag family "MONEY" contains common nouns, proper nouns, adjectives, and adverbs, the semantic component of whose tags within the ATR General English Tagset, is "money": 500-stock, Deposit, TOLL-FREE, inexpensively, etc.

One class of constraints consisted of the presence, within the 6 sentences (from the same document)[52] preceding the current sentence, of one or more instances of a given tag family. This type of constraint came in two varieties: either including, or excluding, the words within the sentence of the word being tagged. Where these intrasentential words were included, they consisted of the set of words preceding the word being tagged, within its sentence.

A second class of constraints added to the requirements of the first class the representation, within the past 6 sentences, of related tag families. Boolean combinations of such events defined this group of constraints. An example is as follows: (a) an instance either of the tag family "person" or of the tag family "personal attribute"(or both) occurs within the 6 sentences preceding the current one; or else (b) an instance of the tag family "person" occurs in the current sentence, to the left of the word being tagged; or, finally, both (a) and (b) occur.

A third class of constraints had to do with the specific word being tagged. In particular, the word being classified is required to belong to a set of words which have been tagged at least once, in the training treebank, with some tag from a particular tag family; and which, further, always shared the same basic syntax in the training data. For instance, consider the words "currency" and "options". Not only have they both been tagged at least once in the training set with some member of the tag family "MONEY" (as well, it happens, as with tags from other tag families);

[52](Black et al., 1998) determined a 6-sentence window to be optimal for this task.

but in addition they both occur in the training set only as nouns. Therefore these two words would occur on a list named "MONEY nouns", and when an instance of either of these words is being tagged, the constraint "MONEY nouns" is satisfied.

A fourth and final class of constraints combines the first or the second class, above, with the third class. E.g. it is both the case that some avatar of the tag family "MONEY" has occurred within the last 6 sentences to the left; and that the word being tagged satisfies the constraint "MONEY nouns". The advantage of this sort of composite constraint is that it is focused, and likely to be helpful when it does occur. The disadvantage is that it is unlikely to occur extremely often. On the other hand, constraints of the first, second, and third classes, above, are more likely to occur, but less focused and therefore less obviously helpful.

4.16 The Experiments

4.16.1 The Four Models

To evaluate the utility of long-range semantic context we performed four separate experiments. All of the models in the experiments include the basic ME tag–n–gram tagger constraints listed in subsection 4.15. The models used in our experiments are as follows:

(1) The first model is a model consisting ONLY of these basic ME tag–n–gram tagger constraints. This model represents the baseline model.

(2) The second model consists of the baseline model together with constraints representing extrasentential tag triggers. This experiment measures the effect of employing the triggers specified in (Black et al., 1998) —i.e. the presence (or absence) in the previous 6 sentences of each tag in the tagset, in turn— to assist a real tagger, as opposed to simply measuring their mutual information. In other words, we are measuring the contribution of this long-range information over and above a model which uses local tag–n–grams as context, rather than measuring the gain over a naive model which does not take context into account, as was the case with the mutual information experiments in (Black et al., 1998).

(3) The third model consists of the baseline model together with the four classes of more sophisticated question-based triggers defined in the previous section.

(4) The fourth model consists of the baseline model together with both the long-range tag trigger constraints and the question-based trigger constraints.

We chose the model underlying a standard tag–n–gram tagger as the baseline because it represents a respectable tagging model which most readers will be familiar with. The ME framework was used to build the models since it provides a principled manner in which to integrate the diverse sources of information needed for these experiments.

4.16.2 Experimental Procedure

The performance of each the tagging models is measured on a 53,000–word test treebank hand-labelled to an accuracy of over 97% (Black et al., 1996; Black et al., 1998). We measure the model performance in terms of the perplexity of the tag being predicted. This measurement gives an indication of how useful the features we supply could be to an n–gram tagger when it consults its

Description	Number
Tag set size	1837
Word vocabulary size	38138
Bigram trigger number	18520
Trigram trigger number	15660
Long history trigger number	15751
Question trigger number	82425

Table 19: Vocabulary sizes and number of triggers used

#	Question Description	MI (bits)
1	Person or personal attribute word in full history	0.024410
2	Word being tagged has taken NN1PERSON in training set	0.024355
3	Person or personal attribute word in remote history	0.024294
4	Person or personal attribute or other related tags in full history	0.020777
5	Person or personal attribute or other related tags in remote history	0.020156

Table 20: The 5 triggers for tag NN1PERSON with the highest MI

model to obtain a probablity distribution over the tagset for a particular word. Since our intention is to gauge the usefulness of long-range context, we measure the performance improvement with respect to correctly (very accurately) labelled context. We chose to do this to isolate the effect of the correct markup of the history on tagging performance (i.e. to measure the performance gain in the absence of noise from the tagging process itself). Earlier experiments using predicted tags in the history showed that at current levels of tagging accuracy for this tagset, these predicted tags yielded very little benefit to a tagging model. However, removing the noise from these tags showed clearly that improvement was possible from this information. As a consequence, we chose to investigate in the absence of noise, so that we could see the utility of exploiting the history when labelled with syntactic/semantic tags.

The resulting measure is an idealization of a component of a real tagging process, and is a measure of the usefulness of knowing the tags in the history. In order to make the comparisons between models fair, we use correctly-labelled history in the n-gram components of our models as well as for the long-range triggers. As a consequence of this, no search is nescessary.

The number of possible triggers is obviously very large and needs to be limited for reasons of practicability. The number of triggers used for these experiments is shown in Table 19. Using these limits we were able to build each model in around one week on a 600MHz DEC-alpha. The constraints were selected by mutual information. Thus, as an example, the 82425 question trigger constraints shown in Table 19 represent the 82425 question trigger constraints with the highest mutual information.

The improved iterative scaling technique (Della Pietra et al., 1997) was used to train the parameters in the ME model.

#	Model	Perplexity	Perplexity Reduction
1	Baseline n-gram model	2.99	0.0%
2	Baseline + long-range tag triggers	2.76	7.6%
3	Baseline + question-based triggers	2.41	19.4%
4	Baseline + all triggers	2.35	21.4%

Table 21: Perplexity of the four models

4.17 The Results

Table 21 shows the perplexity of each of the four models on the testset.

The maximum entropy framework adopted for these experiments virtually guarantees that models which utilize more information will perform as well as or better than models which do not include this extra information. Therefore, it comes as no surprise that all models improve upon the baseline model, since every model effectively includes the baseline model as a component.

However, despite promising results when measuring mutual information gain (Black et al., 1998), the baseline model combined only with extrasentential tag triggers reduced perplexity by just a modest 7.6% . The explanation for this is that the information these triggers provide is already present to some degree in the n–grams of the tagger and is therefore redundant.

In spite of this, when long-range information is captured using more sophisticated, linguistically meaningful questions generated by an expert grammarian (as in experiment 3), the perplexity reduction is a more substantial 19.4%. The explanation for this lies in the fact that these question-based triggers are much more specific. The simple tag-based triggers will be active much more frequently and often inappropriately. The more sophisticated question-based triggers are less of a blunt instrument. As an example, constraints from the fourth class (described in the constraints section of this paper) are likely to only be active for words able to take the particular tag the constraint was designed to apply to. In effect, tuning the ME constraints has recovered much ground lost to the n–grams in the model.

The final experiment shows that using all the triggers reduces perplexity by 21.4%. This is a modest improvement over the results obtained in experiment 3. This suggests that even though this long-range trigger information is less useful, it is still providing some additional information to the more sophisticated question–based triggers.

Table 20 shows the five constraints with the highest mutual information for the tag NN1PERSON (singular common noun of person, e.g. lawyer, friend, niece). All five of these constraints happen to fall within the twenty-five constraints of any type with the highest mutual information with their predicted tags. Within Table 20, "full history" refers to the previous 6 sentences as well as the previous words in the current sentence, while "remote history" indicates only the previous 6 sentences. A "person word" is any word in the tag family "person", hence adjectives, adverbs, and both common and proper nouns of person. Similarly, a "personal attribute word" is any word in the tag family "personal attribute", e.g. left–wing, liberty, courageously.

4.18 Conclusion

Our main concern in this subsection has been to show that extrasentential information can provide significant assistance to a real tagger. There has been almost no research done in this area, possibly due to the fact that, for small syntax–only tagsets, very accurate performance can be obtained

labelling the Wall Street Journal corpus using only local context. In the experiments presented, we have used a much more detailed, semantic and syntactic tagset, on which the performance is much lower. Extrasentential semantic information is needed to disambiguate these tags. We have observed that the simple approach of only using the occurrence of tags in the history as features did not significantly improve performance. However, when more sophisticated questions are employed to mine this long-range contextual information, a more significant contribution to performance is made. This motivates further research toward finding more predictive features. Clearly, the work here has only scratched the surface in terms of the kinds of questions that it is possible to ask of the history. The maximum entropy approach that we have adopted is extremely accommodating in this respect. It is possible to go much further in the direction of querying the historical tag structure. For example, we can, in effect, exploit grammatical relations within previous sentences with an eye to predicting the tags of similarly related words in the current sentence. It is also possible to go even further and exploit the structure of full parses in the history.

5 Predicting Meaning and Function of Phrases and Sentences: Parsing

Note (EB 1999): This subsection presents our approach to parsing, i.e. to finding the correct parse with respect to our grammar of English for any sentence of English. The experimental results presented are roughly current.

5.1 Introduction

This subsection 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., 1993; Grinberg et al., 1995; Schabes, 1992), but arguably at a less detailed and informative level than in the research reported here.

In what follows, subsection 5.2 explains the contribution to the prediction process of the grammar and of the lexical generalizations created by our grammarian. Subsection 5.3 shows, from a formal standpoint, how prediction is carried out, and more generally how the parser operates. Subsection 5.4 presents experimental results. Finally, subsection 5.5 details our efforts to radically expand the size of our training corpus by employing techniques of treebank conversion.

5.2 How the Grammar and Lexical Generalizations Help

5.2.1 How the Grammar Helps

Figure 8 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

```
[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 8: Three ATR/Lancaster English Treebank Sentences: One from Credit Union Brochure, and Two (Non–Sequential) from Chinese Take–Out Food Flier

parser. The Treebank consists of a correct parse for each sentence it contains, with respect to the ATR English Grammar.[53] Every non-terminal node is labelled with the name of the ATR English Grammar rule[54] that generates the node; and each word is labelled with one of the 2843 tags in the Grammar's tagset.[55] 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

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

[54] There are 1155 rules in the Grammar.

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

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 "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.

5.2.2 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.

5.2.3 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, 1996; Ushioda, 1996b).

5.3 How Prediction Is Carried Out

5.3.1 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.[56] 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,[57] 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;
- 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);

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

- 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.

5.3.2 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.

5.4 Experimental Results

5.4.1 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;[58] 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 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

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

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.[59] 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.[60]

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[61] of all tags must be correct.

The average number of different correct "exact syntactic matches"[62] 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.[63] For critiques of other approaches to broad-coverage parser and tagger evaluation, see (Black, 1994b).

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 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.[64] 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

[59] 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.

[60] 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.

[61] and often some of the semantics

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

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

[64] 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.

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 22: 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.

they were parsed and tagged. 76% of these 248 sentence pairs were such exact matches.[65]

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.[66] The result of this scoring was a 8.4% expected parsing error rate.[67]

5.4.2 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.[68] We present parsing results both for text which starts out correctly tagged (Table 22)[69] and for raw text (Table 23). Results for parsing from raw text are given for both the exact–match and exact–syntactic–match criteria described in 5.4.

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 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 23 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%.

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

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

[67] 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.

[68] 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. Futher note (EB 1999): We are currently at about 85% on this metric.

[69] 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 23: 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 24: Comparison of IBM Manuals and ATR/Lancaster General–English Treebanks

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 24 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.[70] Table 4 shows relevant

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

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 25: Parsing results reported by Jelinek et. al. for IBM Manuals task; see Table 24 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 22–23 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.

5.5 Towards Radically Expanding Training–Set Size Via Treebank Conversion

5.5.1 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[71] 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 5.3, 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 roughly 3,000 lexical tags and about 1,100 different non–terminal node labels,[72] as mentioned in 5.2. Figure 9 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

[71] and presumably their source grammar as well

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

```
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 9: IBM/Lancaster Treebank and ATR/Lancaster Parses For Same Sentence

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.

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

$$p(A|F) \propto p(F|A)p(A) \tag{11}$$

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 5.3, 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.

5.5.3 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.[73] 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

[73] Cf. 5.2

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 26: 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.

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.

5.5.4 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. 5.4), 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 26 displays exact–match parsing results for a normal 6,556–word test set[74]. Crucially, the amount of training data here, 118,489 words, is only 17.5% as large as for the models of Tables 22–23. 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,

[74] i.e. not for a "golden standard" test set as described in 5.4, in which all parses are indicated for each test sentence

is to utilize large–scale treebank conversion in the tagging domain to overcome the problem noted in 5.5, 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 described earlier in this section.[75] 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.

6 Application To AI Tasks: Upper–Bound Experimentation

6.1 Introduction

In this section we present two sets of experiments, one in speech recognition and the other in speech synthesis—which represent inquiries into how much help our software could be to these two Artificial Intelligence tasks, assuming for a moment that our predictions were totally accurate. By asking the question in this mode (i.e. as a so–called "upper–bound experiment"), we focus specifically on the value of the information that is delivered by our linguistic analyses, when the right analysis is found. That is, we inquire how valuable our particular way of "milking" the information in text is, *in principle*, for two major applications within Artificial Intelligence. If the answer is that a great deal of value would be contributed, if only our prediction were extremely accurate, then we are justified in continuing our work toward achieving just this degree of accuracy. If not, we may not be so justified. In fact, the results show very clearly the overwhelming benefit to these applications of the information we provide. The bulk of this section details the experimental work on the speech recognition application. At the end of the section, we refer the reader to the original published article presenting the extremely successful work on the speech synthesis application. Taken together, these two sets of experimental results furnish compelling justification for the continuaton of our effort to achieve extremely high prediction accuracy in our parsing and tagging. Below, then, is the original report of these experiments:

It appears intuitively that information from earlier sentences in a document ought to help reduce uncertainty as to the identity of the next word at a given point in the document. (Rosenfeld, 1996) and (Lau et al., 1993) demonstrate a significant "word/word trigger–pair" effect. That is, given that certain "triggering" words have already occurred in a document, the probability of occurrence of specific "triggered" words is raised significantly.

The present section undertakes to demonstrate that semantic/syntactic part–of–speech tags, and parse structure of *previous* sentences of the document being processed, can add trigger information to a standard n–gram language model, over and above the improvement delivered by word/word triggering along the lines of the work by Rosenfeld and Lau et al.[76] We formulate "linguistic–question" triggers which query either: (a) the tags of the words to the left of, and in the same

[75] 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.

[76] (Chelba et al., 1998) explore the problem of utilizing the parse structure of the sentence in which the word to be predicted occurs. The current work can be viewed as complementary to the line of research of Chelba and Jelinek, in that we ignore, to a fair extent, the syntactic structure of the sentence in which the word occurs that is being predicted, and we focus instead on the syntactic and semantic information contained in the sentences prior to the one featuring the word being predicted.

sentence as, the word being predicted; or (b) parse structure and/or tags within any or all of the previous sentences of the document to which the word belongs that is being predicted; or both of (a) and (b) together. Each of these questions then triggers a particular word in the vocabulary, i.e. raises the probability of that word's being the next word of the document.

As the source of both tags and parses in the present experiments, we use a 181,000–word subset of the approximately–1–million–word ATR General English Treebank (Black et al., 1996). This treebank subset consists exclusively of text drawn from Associated Press newswire and Wall Street Journal articles. The 181,000 words are partitioned into a training set of 167,000 words and a test set of 14,000 words. We utilize this portion only of the treebank, as opposed to the entire corpus, in order to match the text type of the raw data set used to train our baseline n–gram language model, which is AP and WSJ text in roughly the same proportions as in our treebank, and of course not including any portion of our training or test text.

We train (i) a baseline 200–million–word n–gram language model; (ii) a model combining this baseline plus a word/word trigger model trained on a 10–million–word subset of the larger training corpus; and finally (iii) a model combining both (i) and (ii) with linguistic–question triggers trained as just indicated. Performance differences of (i/ii/iii) are measured, with the result that model (iii) is shown to yield a significant perplexity reduction vis-a-vis models (i) and (ii).

In what follows, subsection 6.2 provides a basic overview of the language modelling techniques employed; subsection 6.3 discusses and offers examples of the linguistic questions of model (iii); subsection 6.4 describes the language–modelling experiments we performed, and presents our experimental results; and subsection 6.5 discusses our results and indicates future research directions.

6.2 The Language Model (LM)

6.2.1 ME Model

Our language model is a maximum entropy (ME) model of the following form:

$$P(w|h) = \gamma \prod_{k=0}^{K} \alpha_k^{f_k(h,w)} P_b(w|h_0) \tag{12}$$

where:

- w is the word we are predicting;
- h is the history of w;
- γ is a normalization coefficient;
- K is the number of triggers;
- $\alpha_k (k = 0, 1, \cdots, K)$ is the weight of trigger f_k;
- $f_k (i = 0, 1, \cdots, K)$ are trigger functions. $f_k \in \{0, 1\}$;
- $P_b(w|h_0)$ is the base language model.

In our experiments we use as base language models both a conventional trigram model and the extension of this model with long history word triggers. The improved iterative scaling technique (Della Pietra et al., 1997) is used to train the parameters in the ME model.

6.2.2 Trigger selection

The linguistic–question information is embodied in our model in the form of "triggers". A trigger pair $qw = (q, w)$ constists of a triggering question q together with a triggered word w. The number of possible triggers is the product of the number of questions with the number of words in the vocabulary. This gives rise to too many features from which to build an ME model in a reasonable time. We therefore select only those trigger pairs which can be expected to provide the most benefit to the model. We use mutual information (MI) to select the most useful trigger pairs (for more details, see (Rosenfeld, 1996)). That is, we use the following formula to gauge a feature's usefulness to the model:

$$\begin{aligned} MI(q,w) &= P(q,w)\log\frac{P(w|q)}{P(w)} \\ &+ P(q,\overline{w})\log\frac{P(\overline{w}|q)}{P(\overline{w})} \\ &+ P(\overline{q},w)\log\frac{P(w|\overline{q})}{P(w)} \\ &+ P(\overline{q},\overline{w})\log\frac{P(\overline{w}|\overline{q})}{P(\overline{w})} \end{aligned}$$

where:

- w is the word we are predicting;

- q is a triggering feature (e.g. the answer to a linguistic question).

In the final trigger set, we use only those trigger pairs having the highest mutual information.

6.3 Linguistic Information

The experiments reported here consist in adding "linguistic–question constraints"[77] to a baseline n–gram language model. To understand the linguistic questions used, one needs some familiarity with the ATR General English Treebank and the the ATR General English Grammar and Tagset. For detailed presentations, see (Black et al., 1998; Black et al., 1997; Black et al., 1996). Briefly, however, each verb, noun, adjective and adverb in the ATR tagset includes a semantic label, chosen from 42 noun/adjective/adverb categories and 29 verb/verbal categories, some overlap existing between these category sets. Proper nouns, plus certain adjectives and certain numerical expressions, are further categorized via an additional 35 "proper–noun" categories. These semantic categories are intended for any "Standard–American–English" text, in any domain. Sample categories include: "physical.attribute" (nouns/adjectives/adverbs), "alter" (verbs/verbals), "interpersonal.act" (nouns/adjectives/adverbs/verbs/verbals), "orgname" (proper nouns), and "zipcode" (numericals). The semantic categorization is, of course, in addition to an extensive syntactic classification, involving some 165 basic syntactic tags.

The ATR English Grammar is unrestricted in its coverage, and particularly detailed and comprehensive, vis-a-vis other existing grammars. For instance, complete syntactic and semantic analysis is performed on all nominal compounds. Again, see the above–cited references for details.

Each parse of the ATR Treebank was entered by hand by a professional expert in parsing and tagging with the ATR English Grammar (Black et al., 1996). This Treebank is used as training data for an unrestricted–coverage parser of English (Black et al., 1997).

[77] as well as "word/word triggers"

#	Question Description	MI (bits)
1a	Any reference to a female within the last 12 sents of doc	0.001210
2a	Many nouns, adj or adv of verbal action (e.g. statement) within last 100 sents	0.000803
3a	Many nouns, adj or adv of helping (e.g. assistance) within last 100 sents	0.000737
1b	Any subject pronoun to the immediate left	0.000579
2b	Subject of current sentence is a person and verb is likely	0.000407
3b	Many recent sents had person subjects and "saying" main verbs AND Subject of current sentence is a person and verb is likely	0.000314

Table 27: Selected triggers from top–20–highest–MI linguistic–question triggers for the words "Mrs." and "added"

One can get a feel for the type of linguistic–question triggers we defined via Table 27, which shows three triggers with high mutual information with the word "Mrs.", and three for "added". The trigger with the highest mutual information with the word "Mrs." among all linguistic–question triggers does not ask either about tags or parse structure, but simply makes good use, over raw text, of our "Question Language", the flexible language for formulating grammar–based and lexically–based questions about Treebank text, which we normally use to compose contextual questions about text which we are parsing with our probabilistic parser.[78] Specifically, the question, defined over raw text, determines whether any reference has been made to a female, within the last 12 sentences of the current document.

A question which asks about tags is question 2a of Table 27. It queries the semantic portion of tags within the entire history of the document, and determines whether tags have frequently occurred which label nouns, adjectives or adverbs of saying, writing, objecting, or other verbal activities. A "yes" answer to this question turns out to raise the probability of the word "Mrs." as the next word of a document.

Finally, question 3b queries the complex parse structure of previous sentences of the document. The question tests whether frequently in the history of the document, sentences occurred with a human subject and a main verb of verbal activity, e.g. "Mr. Smith stated..." In addition, it tests the current sentence to see whether a human subject has just been received, and a verb now appears to be likely to occur. The expectation, thus, is that a verb of saying will now occur. This expectation turns out to be realized for the verb "added", as there is a relatively high correlation between a "yes" for this question and the occurrence of the word "added".

6.4 The Experiments

6.4.1 Experimental Procedure

We used the well–known trigram LM as the base LM for our experiments. This model was selected because it represents a respectable language model which most readers will be familiar with. The ME framework was used to build the derivative models since it provides a principled manner in which to integrate the diverse sources of information needed for these experiments.

In all models built for these experiments we use a word vocabulary of 20001 (the 20000 most frequent words plus a token for words not in the vocabulary). We used a corpus of newspaper text

[78] For details, see (Black et al., 1997).

Model	Tri20M.k4	Tri100M.k4	Tri200M.k8
unigram	20001	20001	20001
bigram	395663	1230040	1204727
trigram	527782	2724346	2492309

Table 28: Trigram model size varying dataset size

Model	Base PP	Base+Q's	Change(%)
Tri20M.k4	153.0	142.7	6.7
Tri100M.k4	117.8	110.0	6.6
Tri200M.k8	108.0	101.0	6.5

Table 29: Effect of varying dataset size

drawn from 1987–1996 Wall Street Journal and Associated Press Newswire in equal proportion. Certain types of words were mapped to generic tokens representing the class of word. These were: words representing time of day (e.g. 12:21), dates (e.g. 11/02/64), price expressions (e.g. $100) and year expressions (e.g. 1970–1999). The mapping was done using simple regular-expression pattern matching. The substitutions were implemented to assist the trigram model, which is unable to ask questions about the internal structure of words and cannot be expected to form useful n-grams from this class of words. The linguistic questions, however, being able to query the word's internal structure, were more effective on the raw words themselves and were used in that way. The vocabulary, and therefore the words being predicted, was constructed from data in which these tokens had been mapped.

The training set used to train the linguistic question–based triggers for all experiments was approximately 167,000 words of hand–labelled and –parsed ATR treebank, drawn from Wall Street Journal and Associated Press texts. The test set consisted of 14,000 words of hand–labelled and –parsed ATR treebank, again drawn in the same proportion from Wall Street Journal and Associated Press. We measure the test set perplexity (PP) to gauge the quality of the models produced.

6.4.2 Effect of Dataset Size

In this experiment we used base trigram models of three differing sizes. The three models: Tri20M.k4 (k4 = cutoff of 4), Tri100M.k4 and Tri200M.k8 were built from 20M, 100M and 200M words of training data, respectively. Table 28 shows the number of n–grams we used in our models. Table 29 shows the reduction in perplexity. Note that here we used 33000 question–based triggers and the question set size from which the triggers were produced was 396.

In Table 29, "Base" is the perplexity of the base trigram model before any ME training. "Base + Q's" is the perplexity of the full ME model after training. "Change" is the perplexity reduction resulting from using our question triggers.

Notice that increasing the quality of the underlying trigram LM has little effect on the change in perplexity resulting from adding the information from linguistic questions. This indicates that the additional information will be useful to any trigram LM and that simply improving the LM by

Model	PP	Change (%)
Base (Tri200M.k8)	108.0	–
Base + WTModel	94.4	12.6
Base + Q's	95.8	11.3
Base + WTModel + Q's	84.6	21.7

Table 30: The effect of combining the models

adding more data is no substitute for this information.

6.4.3 Effect of Adding Word Triggers

In this experiment we measure the effect of using long–range word triggers on our corpus together with the effect of combining these with our question–based triggers. 39367 long history word triggers are chosen by mutual information from 200 million words of data. Due to the prohibitively long training times needed to train models using word triggers we restricted the training set for the ME training to 10M words. The base language model was trained on the full 200M word corpus. We then used the ME model built by adding word–triggers to the base model as the base model for a second ME model which incorporated our question–based triggers. We found this approach effective in dealing with the large number of triggers involved. The number of question–based triggers used was 110,000 and the question set size from which the triggers were produced was 6,659. The results are shown in Table 30.

6.5 Discussion

The maximum entropy framework adopted for these experiments virtually guarantees that models which utilize more information will perform as well as or better than models which do not include this extra information. Therefore, it comes as no surprise that all models improve upon the baseline model, since every model effectively includes the baseline model as a component. The experiments presented here have focused on showing that that we can glean useful information from the parse structure and part–of–speech tags in the history of the word being predicted. Our main result is that this information is useful, and is of similar magnitude to that provided by the long–range word triggers used by (Rosenfeld, 1996). Moreover, when these triggers are used in conjunction with a model incorporating long–range word triggers, almost all of the perplexity gain is inherited by the new model. This indicates that the information we are providing is largely new and complementary. This is in line with our intuition, given the nature of the questions we ask. Furthermore, we obtained this gain from a very small 167,000 word training corpus (as opposed to the 10 million word corpus used to train the long–range word triggers). It is reasonable to expect significant improvement on domains where more data is available to train from.

This work is a first attempt at exploiting the parse structure in the extrasentential history to assist a language model. A major practical concern is that the predictions are being made from correctly analysed text rather than the output of a parsing device. Our intention in this paper was to show that there is useful information in the parses in the history. In further research, we intend to incorporate a real parsing device.

When a real parser is used, the system (including the grammarian writing the questions) will need to overcome the errors made by the parser/tagger. However, one point in favour of this approach is that if we train from the output of the parser (one way to learn to predict from only the reliable parts of the parse), we will have a much larger corpus from which to train the question–based component of the LM. Additionally, although we are currently able to ask quite sophisticated questions of the structure of parses in the history, we feel that we can realize considerable gain by further developing the language we are using to ask these questions, and thereby improving their expressive power.

6.6 Application To Speech Synthesis Tasks

The reader is referred to (Campbell et al., 1997) for an account of a second upper–bound experiment aimed at quantifying the contribution of the English–text–analysis software being described here. In this work, the aim is to measure the diminution in error rate, due to our language–analysis software, of a state–of–the–art speech synthesis device, when use is made of the linguistic information which we deliver, about text for "reading" by the device. The results reported in (Campbell et al., 1997) for this upper–bound experiment are that error rate for pause insertion is reduced by 56% and error rate for prominence assignment is reduced by 62%. Thus there appears to be a huge potential for our software to assist in the task of speech synthesis, once our accuracy can be brought to a level approaching that of human experts at the task of assigning language structure using our approach, and once speed can be sufficiently augmented.

References

L. Bahl, F. Jelinek, R. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, PAMI–5, 2:179–190.

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

D. Beeferman, A. Berger, and J. Lafferty. 1997. A Model of Lexical Attraction and Repulsion. In *Proceedings of the ACL-EACL'97 Joint Conference*, Madrid.

E. Black, A. Finch, H. Kashioka. 1998. Trigger-Pair Predictors in Parsing and Tagging. In *Proceedings, 36th Annual Meeting of the Association for Computational Linguistics, 17th Annual Conference on Computational Linguistics*, pages 131–137, Montreal.

E. Black, S. Eubank, H. Kashioka, J. Saia. 1998. Reinventing Part-of-Speech Tagging. *Journal of Natural Language Processing (Japan)*, 5:1.

E. Black, S. Eubank, H. Kashioka. 1997. Probabilistic Parsing of Unrestricted English Text, With A Highly-Detailed Grammar. In *Proceedings, Fifth Workshop on Very Large Corpora*, Beijing/Hong Kong.

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 COLING 96*, pages 107–112, Copenhagen.

E. Black. 1994a. An experiment in customizing the Lancaster Treebank. In Oostdijk and de Haan, 1994, pages 159-168.

E. Black. 1994b. A New Approach to Evaluating Broad-Coverage Parser/Grammars. In Proceedings, International Conference on New Methods in Language Processing. Manchester, UK.

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. Black, F. Jelinek, J. Lafferty, R. Mercer, S. Roukos. 1992. Decision tree models applied to the labelling of text with parts–of–speech. In *Proceedings, DARPA Speech and Natural Language Workshop*, Arden House, Morgan Kaufman Publishers.

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

E. Brill. 1993. Automatic grammar induction and parsing free text: A Transformation–based approach. In *Proceedings, 31st Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics.

E. Brill. 1994. Some Advances in Transformation–Based Part of Speech Tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 722-727, Seattle, Washington. American Association for Artificial Intelligence.

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.

N. Campbell, T. Hebert, E. Black. 1997. Parsers, Prominence and Pauses. In *Proceedings, Eurospeech 97*, Greece.

C. Chelba, F. Jelinek. 1998. Exploiting Syntactic Structure for Language Modelling. In *Proceedings of COLING–ACL 98*, Montreal, pages 225-231.

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.

S. Della Pietra, V. Della Pietra, J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

S. Della Pietra, V. Della Pietra, R. Mercer, S. Roukos. 1992. Adaptive language modeling using minimum discriminant information. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, I:633–636.

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.

X. Huang, F. Alleva, H.–W. Hon, M.–Y. Hwang, K.–F. Lee, and R. Rosenfeld. 1993. The SPHINX–II speech recognition system: an overview. *Computer Speech and Language*, 2:137–148.

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, ARPA Workshop on Human Language Technology*, pages 260-265, Plainsboro, New Jersey, ARPA.

F. Jelinek and R. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition In Practice*, E. S. Gelsema and N. L. Kanal, eds., pages 381–402, Amsterdam: North Holland.

F. Jelinek, R. L. Mercer, L. R. Bahl, J. K. Baker. 1977. Perplexity—a measure of difficulty of speech recognition tasks. In *Proceedings of the 94th Meeting of the Acoustic Society of America*, Miami Beach, FL.

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

F. Karlsson, A. Voutilainen, J. Heikkila, and A. Anttila. 1995. Constraint Grammar: A Language–Independent System for Parsing Unrestricted Text. Mouton de Gruyter: Berlin and New York.

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

E. Kruisinga. 1931. *A Handbook Of Present–Day English.* P. Noordhoff. Groningen.

H. Kucera and W. N. Francis. 1967. *Computational Analysis of Present–Day American English.* Brown University Press. Providence, RI.

R. Kuhn, R. De Mori. 1990. A Cache–Based Natural Language Model for Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583.

J. Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. In *Computer Speech and Language*, 6:225–242.

J. Kupiec. 1989. Probabilistic models of short and long distance word dependencies in running text. In *Proceedings of the DARPA Workshop on Speech and Natural Language*, pages 290–295.

R. Lau. 1994. Adaptive Statistical Language Modelling. *Master's Thesis*, Massachusetts Institute of Technology, MA.

R. Lau, R. Rosenfeld, S. Roukos. 1993. Trigger-based language models: a maximum entropy approach. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, II:45–48.

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

D. M. Magerman. 1995. Statistical Decision–Tree Models for Parsing. In *Proceedings, 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, Massachusetts, Association for Computational Linguistics.

D. Magerman. 1994. *Natural Language Parsing As Statistical Pattern Recognition.* Ph.D. Thesis, Stanford University.

D. M. Magerman and M. P. Marcus. 1991. Pearl: A Probabilistic Chart Parser. In *Proceedings, European ACL Conference*, March 1991, Berlin, Germany.

M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. *Proceedings, ARPA Human Language Technology Workshop*, Morgan Kaufmann Publishers Inc., San Francisco.

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.

L. Marquez and L. Padro. 1997. A Flexible POS Tagger Using An Automatically Acquired Language Model. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 238–245, Madrid.

B. Merialdo 1994. Tagging English text with a probabilistic model. In *Computational Linguistcs*, 20(2):155–172..

N. Oostdijk and P. de Haan, Editors. 1994. *Corpus-Based Research Into Language: In honour of Jan Aarts.* Rodopi Editions. Amsterdam.

D. Paul. 1990. Algorithms for an optimal a^* search and linearizing the search in th e stack decoder. *Proceedings of the June 1990 DARPA Speech and Natural Language Work shop.*

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

A. Ratnaparkhi. 1997. A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. In *Proceedings, Second Conference on Empirical Methods in Natural Language Processing*, Providence, RI.

A. Ratnaparkhi. 1996. A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania.

R. Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, 10:187–228.

G. Sampson. 1994. *English for the Computer.* Oxford, Oxford University Press.

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 Nonterminals. 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. *Proceedings, COLING 96, 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.

H. van Halteren and T. van den Heuvel. 1990. *Linguistic Exploitation of Syntactic Databases.* Rodopi Editions. Amsterdam.

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.

R. Weischedel, M. Meteer, R. Schwartz, L. Ramshaw, and J. Palmucci. 1993. Coping with Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics*, 19.2:359-382.