Machine Learning for Information Extraction in Informal Domains

Dayne Freitag

November, 1998

CMU-CS-99-104

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:

Tom Mitchell, Chair
Jaime Carbonell
David Evans
Oren Etzioni, University of Washington

This research was sponsored by Wright Laboratory, Aeronautical Systems Center under grant number F33615-93-1-1330 and Rome Laboratory under grant number F30602-97-1-0215, both of the Air Force Materiel Command-USAF, and by the Defense Advanced Research Projects Agency (DARPA). Part of this research was conducted during a summer internship at Justsystem Pittsburgh Research Center.

Keywords: machine learning, information extraction, information retrieval, multistrategy learning

Abstract

Information extraction, the problem of generating structured summaries of human-oriented text documents, has been studied for over a decade now, but the primary emphasis has been on document collections characterized by well-formed prose (e.g., newswire articles). Solutions have often involved the hand-tuning of general natural language processing systems to a particular domain. However, such solutions may be difficult to apply to "informal" domains, domains based on genres characterized by syntactically unparsable text and frequent out-of-lexicon terms. With the growth of the Internet, such genres, which include email messages, newsgroup posts, and Web pages, are particularly abundant, and there is no lack of potential information extraction applications. Examples include a program to extract names from personal home pages, or a system that monitors newsgroups where computers are offered for sale in search of one that matches a user's specifications.

This thesis asks whether it is possible to design general-purpose *machine learning* algorithms for such domains. Rather than spend weeks or months manually adapting an information extraction system to a new domain, we would like a system we can train on some sample documents and expect to do a reasonable job of extracting information from new ones. This thesis poses the following questions: What sorts of machine learning algorithms are suitable for this problem? What kinds of information might a learner exploit in an informal domain? Is there a way to combine heterogeneous learners for improved performance?

This thesis presents four learners representative of a diverse set of machine learning paradigms—a rote learner (Rote), a statistical term-space learner based on the Naive Bayes algorithm (BayesIDF), a hybrid of BayesIDF and the grammatical inference algorithm Alergia (BayesGI), and a relational learner (SRV). It describes experiments testing these learners on three different document collections—electronic seminar announcements, newswire articles describing corporate acquisitions, and the home pages of courses and research projects at four large computer science departments. Finally, it describes a modular *multistrategy* approach which arbitrates among the individual learners, using regression to re-rank learners' predictions and achieve performance superior to that of the best individual learner on a problem.

Contents

1 Introduction **1**

1.1 Background . . . 2
1.2 Point of Departure . . . 4
1.2.1 Study in Diverse Domains . . . 5
1.2.2 Comparison of Multiple Learners . . . 5
1.2.3 Investigation of Multistrategy Learning . . . 8
1.3 Claims . . . 8
1.4 Thesis Organization . . . 9

2 The Problem Space **11**

2.1 Problem Definition . . . 11
2.1.1 Formal Framework . . . 12
2.1.2 Discussion . . . 12
2.2 Document Views . . . 13
2.2.1 The Terms View . . . 13
2.2.2 The Mark-Up View . . . 13
2.2.3 The Layout View . . . 16
2.2.4 The Typographic View . . . 17
2.2.5 The Linguistic View . . . 18
2.3 Evaluating Performance . . . 19
2.3.1 Unit of Performance . . . 19
2.3.2 Document Outcomes . . . 19
2.3.3 Fragment Outcomes . . . 20
2.3.4 Precision and Recall . . . 21
2.3.5 Problem Difficulty . . . 23
2.4 Domains . . . 25
2.5 MUC . . . 26

3 Term-Space Learning for Information Extraction **29**
3.1 Rote Learning 30
3.2 Naive Bayes 32
3.2.1 Fragments as Hypotheses 33
3.2.2 Derivation of Bayes 34
3.2.3 Modifications 38
3.3 Experiments 40
3.3.1 Case Study: Seminar Announcements 41
3.3.2 Case Study: Acquisitions 45
3.4 Discussion 51

4 Learning Field Structure with GI **55**
4.1 Grammatical Inference 58
4.1.1 General Setting 59
4.1.2 State-Merging Methods 59
4.1.3 Alergia 61
4.2 Inferring Transducers 62
4.3 Experiments 66
4.4 Discussion 72

5 Relational Learning for Information Extraction **73**
5.1 SRV 74
5.1.1 Example Space 75
5.1.2 Features 76
5.1.3 Rule Construction 77
5.1.4 An Example 81
5.1.5 Rule Accuracy Estimation 86
5.1.6 Implementation 86
5.1.7 Time Complexity 91
5.2 Experiments 93
5.2.1 Case Study: Seminar Announcements 94
5.2.2 Case Study: Web Pages 97
5.2.3 Case Study: Newswire Articles 101
5.3 Discussion 107

6 Multistrategy Approaches **119**
6.1 Opportunity . . . 120
6.2 Combining Methods . . . 121
6.2.1 Basic Scheme . . . 122
6.2.2 Regression to Estimate Correctness . . . 123
6.2.3 Bayesian Prediction Combination . . . 124
6.3 Experiments . . . 126
6.4 Discussion . . . 128
6.4.1 Favorable Factors . . . 128
6.4.2 Representation and Learning Paradigm . . . 129

7 Related Work **139**
7.1 Term-Space Learning . . . 139
7.2 Grammatical Inference . . . 140
7.3 Relational Learning . . . 142
7.4 Multistrategy Learning . . . 143

8 Conclusion **145**
8.1 Contributions . . . 146
8.1.1 Informal Domains . . . 146
8.1.2 Comparison of Learning Methods . . . 147
8.1.3 Multistrategy Learning . . . 148
8.2 Insights Gained . . . 148
8.3 Open Questions . . . 150
8.3.1 Grammatical Inference over Feature Vectors . . . 151
8.3.2 Exploiting Field Co-occurrence . . . 152
8.3.3 Using Linguistic Information . . . 153
8.3.4 Using Layout . . . 154
8.3.5 Information Extraction as Navigation . . . 155

A Domains **157**
A.1 Seminar Announcements . . . 157
A.2 Newswire Articles on Acquisitions . . . 158
A.3 University Web Pages . . . 161
A.3.1 Course Pages . . . 161
A.3.2 Research Project Pages . . . 163

B Excerpts **165**
B.1 Seminar Announcements . . . 165
B.2 Acquisitions Articles . . . 169
B.3 University Web Pages . . . 173

C The Tokenizing Library **179**

Bibliography **183**

List of Tables

1.1 Overview of the learners described in this thesis. 6

3.1 Procedure for finding an entry in Rote's dictionary. 31
3.2 The training procedure used by all Bayes variants. 36
3.3 Bayes's estimating procedure for text fragments. 37
3.4 A sample Bayes fragment likelihood estimation for a location phrase (“`Baker Hall Adamson Wing`”) taken from the seminar announcement collection. 38
3.5 BayesLN's estimating procedure for text fragments. 39
3.6 BayesIDF's estimating procedure for text fragments. 40
3.7 A sample BayesIDF fragment likelihood estimation for a location phrase (“`Baker Hall Adamson Wing`”) taken from the seminar announcement collection. 41
3.8 Precision and recall of Rote and three variants of Bayes on the four seminar announcement fields. 42
3.9 Precision at the approximate 25% recall level of Rote and the three variants of Bayes on the four seminar announcement fields. 42
3.10 Peak F1 scores and corresponding precision and recall for Rote and BayesIDF on the seminar announcement fields. 42
3.11 Precision of Rote and BayesIDF on the ten acquisitions fields at two recall levels, 25% and full. 47
3.12 Peak F1 scores, with corresponding precision and recall, for Rote and BayesIDF on the acquisitions fields. 47
3.13 The first three lines of an acquisition article showing a typical pattern of field instantiation: *purchaser* immediately following the dateline. 49

4.1 I/O behavior of transducer induction. 63
4.2 Excerpt from one decision list inferred for the location field. 64
4.3 The features used for inferring alphabet transducers. 64
4.4 The covering procedure used to construct alphabet transducers. 65
4.5 Precision/recall results for Alergia and BayesGI on the *speaker* field, with the alphabet transducer produced using m-estimates, at various settings of Alergia's generalization parameter. 67

4.6 Precision/recall results for BayesIDF, Alergia, and BayesGI on the *location* field, using the m-estimates alphabet transducer, at various settings of Alergia's generalization parameter. 67
4.7 Precision results on the *speaker* field for canonical acceptors (with and without BayesIDF) using five different alphabets. 68
4.8 Precision results on the *location* field for canonical acceptors (with and without BayesIDF) using five different alphabets. 68
4.9 Average size of decision lists generated using information gain and m-estimate metrics across the four seminar announcement fields. 69
4.10 Peak F1 scores for Alergia, BayesIDF, and BayesGI on the seminar announcement fields. 69

5.1 An excerpt from the header of a seminar announcement. 81
5.2 SRV's rule-growing algorithm in pseudocode. 87
5.3 SRV's procedure for finding all but the first literal in a rule. 88
5.4 SRV's procedure for finding the first literal in a rule. 90
5.5 SRV's default features. 94
5.6 Peak F1 scores, with corresponding precision and recall, for all methods on all seminar announcement fields. 97
5.7 Precision and recall of all methods on all seminar announcement fields. . . 97
5.8 HTML features added to SRV's default feature set. Features in italics are relational. 98
5.9 Peak F1 scores of three learners on the three "one-per-document" fields from the Web domain. 100
5.10 Precision and recall of all learners, including SRV with HTML features, on the three OPD fields of the WebKB domain. 100
5.11 Peak F1 scores of all learners, including SRV with HTML features, on the two "many-per-document" fields from the WebKB domain. 100
5.12 Precision and recall of all learners, including SRV with HTML features, on the "many-per-document" fields of the WebKB domain. 101
5.13 Peak F1 scores, and corresponding precision and recall, of Rote, BayesIDF, SRV, and SRV augmented with linguistic features on nine of the acquisitions fields. 104
5.14 Precision and recall at full recall of Rote, BayesIDF, SRV, and SRV augmented with linguistic features on nine of the acquisitions fields. 104
5.15 Precision and recall results from a three-fold experiment on four fields for the three basic learners, plus SRV with syntactic and lexical information (SRV (ling)), SRV with only syntactic information (SRV (lg)), and SRV with only lexical information (SRV (wn)). 107

6.1 Outcome contingency table for the *speaker* field showing the probability that a row learner handled a document correctly, given that a column learner handled it correctly. 120
6.2 The function used by CProb to process a test document. 123
6.3 The function used by CBayes to process a test document. 125
6.4 Peak F1 scores of the multistrategy approach compared with that of the best individual learner (SRV in all cases, unless marked (*) for BayesGI or (**) for Rote). 127

A.1 Corpus statistics for the seminar announcement domain. 158
A.2 Field statistics for the seminar announcement domain. 158
A.3 Corpus statistics for the acquisitions domain. 160
A.4 Field statistics for the seminar announcement domain. 161
A.5 Corpus statistics for the WebKB course pages sub-domain. 162
A.6 Field statistics for the WebKB course pages sub-domain. 162
A.7 Corpus statistics for the WebKB project pages sub-domain. 162
A.8 Field statistics for the WebKB project pages sub-domain. 163

B.1 A complete seminar announcement illustrating the common use of the label/colon device. 166
B.2 A complete seminar announcement illustrating the use of a single short paragraph to convey essential details. 166
B.3 A complete seminar announcement illustrating the mixing of prose with other devices and the use of centering. 167
B.4 A complete seminar announcement illustrating how itemizations, italics, and headlines are emulated. 168
B.5 A complete seminar announcement illustrating the use of an ad hoc table. . 169
B.6 A typical short article from the acquisitions domain. 169
B.7 A complete acquisitions article, the subject of which is not directly a proposed or completed acquisition but a byproduct of one. 170
B.8 A complete acquisitions article in which many details, including the buyer, are not listed. 170
B.9 A complete acquisitions article illustrating some of the subtleties involved in distinguishing the roles of companies. 171
B.10 Another acquisitions article in which the parties are identifiable but difficult to extract. 171
B.11 Beginning of an acquisitions article in which one of the main parties is not mentioned in the first paragraph. 172
B.12 A complete acquisitions article in which one of the main parties is an individual, rather than a company. 173
B.13 The top of a typical course page from the WebKB domain. 174

B.14 A course page excerpt in which instances of *crsInst* are listed linearly. . . . 174
B.15 A course page excerpt in which a HTML table is used to present instances of *crsInst*. 175
B.16 Project page excerpt showing a typical listing of members. 176
B.17 Top of a project page showing instances of *projTitle*. 176
B.18 Top of a project page illustrating instances of *projTitle* in various contexts. . 177

C.1 Excerpt from a seminar announcement showing annotation used to identify field instances. 180

List of Figures

2.1 A seminar announcement. 14
2.2 A seminar announcement as a sequence of literal terms. 14
2.3 Part of a personal home page from the World Wide Web. 15
2.4 A mark-up view of the excerpt shown in Figure 2.3, in which non-markup, non-whitespace characters have been replaced by asterisks. 15
2.5 A layout view of the document shown in Figure 2.1, in which non-whitespace characters have been replaced by asterisks. 16
2.6 A layout view augmented with typographic information. 17
2.7 A syntactic view of the title of the seminar announced in Figure 2.1. 18
2.8 A semantics view, produced using Wordnet, of the title of the seminar announced in Figure 2.1. 18
2.9 A precision/recall graph. 22
2.10 The generic information extraction processing pipeline, according to Cardie. 27

3.1 A hypothetical insertion of a seminar location instance into the discrimination net used to implement Rote's dictionary. 31
3.2 A depiction of the histogram used by Bayes to estimate the position likelihood of test instances. 34
3.3 Precision of Rote and BayesIDF as a function of recall on the seminar *location* field. 43
3.4 Precision of Rote and BayesIDF as a function of recall on the seminar *etime* field. 44
3.5 Precision of Rote and BayesIDF as a function of recall on the *stime* field. . 45
3.6 Precision of Rote and BayesIDF as a function of recall on the seminar *speaker* field. 46
3.7 Precision of Rote and BayesIDF as a function of recall on the *acquired* (purchased company or resource) field. 48
3.8 Precision of Rote and BayesIDF as a function of recall on the *purchaser* field. 49
3.9 Precision of Rote and BayesIDF as a function of recall on the *acqabr* field (short version of *acquired*). 50

3.10 Precision of Rote and BayesIDF as a function of recall on the *dlramt* field. 50
3.11 Precision of Rote and BayesIDF as a function of recall on the *status* field. . 51

4.1 Examples of poor alignment from actual tests of BayesIDF. 56
4.2 Effect of changing criterion of correctness on BayesIDF performance on the *speaker* field. 56
4.3 Effect of changing criterion of correctness on BayesIDF performance on the *location* field. 57
4.4 A canonical acceptor (prefix-tree grammar) representing the training sample $\{110, \lambda, \lambda, \lambda, 0, \lambda, 00, 00, \lambda, \lambda, \lambda, 10110, \lambda, \lambda, 100\}$. 60
4.5 The grammar after merging the states of the grammar shown in Figure 4.4 using Alergia at a particular setting of its generalization parameter α. 60
4.6 The pipeline through which raw text fragments are passed to produce structure estimates. 63
4.7 A small piece of an automaton, used for recognizing seminar locations, learned by Alergia using a decision list created with m-estimates. 63
4.8 Precision/recall plot comparing BayesGI, BayesIDF, and the canonical acceptor (CA grammar) on *speaker*. 70
4.9 Precision/recall plot comparing BayesGI, BayesIDF, and the canonical acceptor (CA grammar) on *location*. 70
4.10 Precision/recall plot comparing BayesGI, BayesIDF, and the canonical acceptor (CA grammar) on *stime*. 71
4.11 Precision/recall plot comparing BayesGI, BayesIDF, and the canonical acceptor (CA grammar) on *etime*. 71

5.1 A text fragment and some of the examples it generates—one positive example, and many negative ones. 75
5.2 Some start times that match the learned rule (carriage returns removed but other whitespace preserved). 85
5.3 A rule learned by SRV to recognize instances of the *speaker* field, its first-order logic equivalent, its English translation, and a fragment of text it matches. 95
5.4 A rule learned by SRV to recognize instances of *location*, its equivalent in first-order logic, its English translation, and a matching fragment with variable bindings. 95
5.5 A rule learned by SRV to recognize instances of *stime*, its equivalent in first-order logic, its English translation, and a matching fragment. 96
5.6 A rule learned by SRV to recognize instances of *etime*, its equivalent in first-order logic, and a matching fragment. 96
5.7 An example of link grammar feature derivation. 102
5.8 One sense of the word "acquisition" and all its generalizations in Wordnet. . 103

5.9 A learned rule for *acqabr* that uses linguistic features, along with two fragments of matching text and relevant linguistic information. 106
5.10 Precision/recall plot comparing all four learners on the *speaker* field from the seminar announcement domain. 108
5.11 Precision/recall plot comparing all four learners on the *location* field from the seminar announcement domain. 109
5.12 Precision/recall plot comparing all four learners on the *stime* field from the seminar announcement domain. 109
5.13 Precision/recall plot comparing all four learners on the *etime* field from the seminar announcement domain. 110
5.14 Precision/recall plot comparing all four learners (include SRV without HTML features) on the *crsNumber* field from the WebKB domain. 111
5.15 Precision/recall plot comparing all four learners (including SRV without HTML features) on the *crsTitle* field from the WebKB domain. 111
5.16 Precision/recall plot comparing all four learners (including SRV without HTML features) on the *projTitle* field from the WebKB domain. 112
5.17 Precision/recall plot comparing all four learners (including SRV without HTML features) on the *crsInst* field (a "many-per-document" field) from the WebKB domain. 112
5.18 Precision/recall plot comparing all four learners (including SRV without HTML features) on the *projMember* field (a "many-per-document" field") from the WebKB domain. 113
5.19 Precision/recall plot comparing all four learners on the *acquired* field for the acquisitions domain. 113
5.20 Precision/recall plot comparing all four learners on the *purchaser* field for the acquisitions domain. 114
5.21 Precision/recall plot comparing all four learners on the *seller* field for the acquisitions domain. 114
5.22 Precision/recall plot comparing all four learners on the *acqabr* field for the acquisitions domain. 115
5.23 Precision/recall plot comparing all four learners on the *purchabr* field for the acquisitions domain. 115
5.24 Precision/recall plot comparing all four learners on the *sellerabr* field for the acquisitions domain. 116
5.25 Precision/recall plot comparing all four learners on the *acqloc* field for the acquisitions domain. 116
5.26 Precision/recall plot comparing all four learners on the *dlramt* field for the acquisitions domain. 117
5.27 Precision/recall plot comparing all four learners on the *status* field for the acquisitions domain. 117

6.1 Combining predictions of different learners for a hypothetical *location* fragment. . . . 121
6.2 The basic combination scheme. . . . 122
6.3 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *speaker* field from the seminar announcement domain. . . . 130
6.4 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *location* field from the seminar announcement domain. . . . 131
6.5 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *stime* field from the seminar announcement domain. . . . 131
6.6 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *etime* field from the seminar announcement domain. . . . 132
6.7 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *acquired* field from the acquisitions domain. 132
6.8 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *purchaser* field from the acquisitions domain. . . . 133
6.9 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *acqabr* field from the acquisitions domain. 133
6.10 Precision/recall plot comparing the best individual learner (BayesGI) with the three combining methods on the *dlramt* field from the acquisitions domain. . . . 134
6.11 Precision/recall plot comparing the best individual learner (Rote) with the three combining methods on the *status* field from the acquisitions domain. . 134
6.12 Precision/recall plot comparing the best individual learner (BayesGI) with the three combining methods on the *crsNumber* field from the WebKB domain. . . . 135
6.13 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *crsTitle* field from the WebKB domain. . . 135
6.14 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *projTitle* field from the WebKB domain. . . 136
6.15 Precision/recall plot comparing the best individual learner (BayesGI) with the three combining methods on the *crsInst* field from the WebKB domain. 136
6.16 Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *projMember* field from the WebKB domain. 137

A.1 Fields defined for the acquisitions domain. . . . 159

Acknowledgments

First things first. My wife, Britt, has played a critical role in the completion of this thesis and my graduate school career. She has rendered assistance in any number of forms during our stay in Pittsburgh—love, encouragement, abiding faith in my abilities, welcome diversion, not to mention financial support. She was party to the decision to play the graduate school gambit in the first place, way back in the days when this meant undergraduate re-education for me, the outcome of which was far from clear. And without her willingness to tear up roots from a community she loved and cross the country to settle in a strange city, this dissertation would never have been possible. Our son, Alex, came onto the scene later in the course of my doctoral work. Although he did not contribute consciously, his very presence served as an invaluable source of perspective, and his tolerant, cheerful disposition (I can claim no credit for this!) made him a welcome addition. Together, Britt and Alex constitute a domestic universe conducive to good, imaginative research.

As I have not hesitated to tell new graduate students, Tom Mitchell is a great advisor. Throughout my time at CMU, he was the source of any number of appealing research ideas, many of which subsequently bore fruit, some of which I had the opportunity to realize and extend. “Opportunity” is a good word to use in describing Tom, whose advisory style involves treating his advisees as colleagues from Day One, spreading before them a wealth of interests, projects, and directions, and inviting them to find their niche. In other words, he gives his students opportunity, never grief. But his true strength as an advisor did not become completely clear to me until my thesis was in the home stretch, when in spite of a schedule that would hobble most people, he carefully reviewed and criticized each of the chapters. The result is a better, more precise presentation.

The other members of my thesis committee—Jaime Carbonell, David Evans, and Oren Etzioni—contributed, at various points in the process, interesting research ideas, complementary perspectives, and suggestions for improving the presentation of ideas. I thank them for their participation.

Throughout my time at CMU, my research trajectory was strongly influenced by certain key collaborators and co-authors. I list them here in the order in which I made their acquaintance: David “Stork” Zabowski, Siegfried Bocionek, Rich Caruana, Thorsten Joachims, Andrew McCallum, and Mark Craven. I learned a lot by watching these people, all good researchers, attack hard, important problems.

Although I entered graduate school well after I was independently established in the world, I would be negligent not to acknowledge my parents’ contribution to my success as a graduate student and a person. Among other things, they are attentive parents and very smart people. Consequently, nature and nurture combined to provide me with the mental faculties and habits of critical thinking that make the exercise of research possible. If I had listened to them early in my college career (“Dayne, you should concentrate on math and science”), I might have reached this point sooner. Instead, I studied literature. I don’t regret the detour, but I do acknowledge my parents’ insightfulness.

A large fraction of a graduate student’s waking hours are spent in the company of a few key people who, as much as anyone else, determine his quality of life—office mates. I have had some good ones: Shumeet Baluja, Geoff Gordon, Jürgen Dingel, Phoebe Sengers, and Belinda Thom. During most of my work on the dissertation, I shared an office with Phoebe and Belinda, who made available to me a whole side of the graduate school experience that otherwise would have been lost on me. They also served as a source of inspiration, both of them persevering and succeeding in the face of great difficulties.

Of course, the list doesn't stop here. And of course, I am bound to omit someone as deserving of acknowledgment as anyone else. Nevertheless, here are the names, in no particular order, of some more people who, wittingly or not, helped me in one way or another: fellow students Justin Boyan, Joseph O'Sullivan, Sèan Slattery, Rosie Jones, and Kamal Nigam; faculty members Scott Fahlman and Yiming Yang; staff members Jean Harpley, Sharon Burks, and Catherine Copetas; and intellectual nomad Johan Kumlien.

Chapter 1

Introduction

> *Information extraction* is the problem of generating stereotypic summaries from free text. Traditional information extraction is performed on journalistic or technical documents and typically involves some linguistic pre-processing. In many domains, however, linguistic processing is difficult, if not impossible. We would like to design a machine learning system that operates in such domains, in addition to more traditional ones. Such a system should exploit sources of information such as term frequency statistics, typography, orthography, meta-text (mark-up), and formatting. As a means of investigating the usefulness of such information, this thesis presents four machine learning algorithms from diverse paradigms and studies their performance on several different information extraction domains. Experiments show it is possible to design algorithms that learn to perform extraction competently in the absence of linguistic information. Further experiments demonstrate that by combining *multiple* learners an even higher level of competence can be achieved.

If I were in the market for a bargain computer, then I would benefit from a system that monitors newsgroups where computers are offered for sale until it finds a suitable one for me. As a critical component of this system I would need a program that converts the information in a single newsgroup post into machine-usable form. An individual summary produced by my program might take the form of a template with typed slots, each of which is filled by a fragment of text from the document (e.g., type: "Pentium"; speed: "200 MHz."; disksize: "3 Gig"; etc.). The design of such a program is essentially an *information extraction* problem. We know what each document in these newsgroups says in general terms; it describes a computer. Information extraction is the problem of extracting the essential details particular to a given document.

Existing work in information extraction can give us some good ideas about how this program should be constructed, but we will find large portions of it inapplicable. Most of this work assumes that we can perform syntactic and semantic processing of a document. Unfortunately, not only do we find strange, syntactically intractable constructions like news headers and user signatures in news posts, but sometimes even the body of a message lacks a single grammatical construct. How should my program handle the "messy" text it is likely to encounter? How can it exploit whatever conventions of presentation are typical of postings for this newsgroup? More interestingly, are there general machine learning methods we can use to *train* a program for use in this and similarly informal domains?

My research addresses this question. I am interested in designing machine learning components for information extraction which are as flexible as possible, which can exploit syntactic and semantic information when it is available, but which do not depend on its availability. Other sources of useful information include:

- Term frequency statistics
- Typography (e.g., capitalization patterns)
- Meta-text, such as HTML tags
- Formatting and layout

The central thesis of this dissertation is that *we can design general-purpose machine learning algorithms that exploit these non-linguistic sources of information, enough for competent performance in many domains, and that by combining learners with different strengths and weaknesses we can realize even better information extraction performance.*

1.1 Background

One of the grand challenges of computer science, in which information extraction plays a part, is the development of automatic methods for the *management* of text, rather than just its transmission, storage, or display. Efforts to meet this challenge are nearly as old as computer science itself. The decades-old discipline of *information retrieval* has developed automatic methods, typically of a statistical flavor, for indexing large document collections and classifying documents. The complementary endeavor of *natural language processing* has sought to model human language processing—with some success, but also with a hard-won appreciation of the magnitude of the task.

The much younger field of information extraction lies somewhere in between these two older endeavors in terms of both difficulty and emphasis. Information extraction can be regarded as a kind of limited, directed natural language understanding. It assumes the existence of a set of documents from a limited domain of discourse, in which each document describes one or more entities or events that are similar to those described in other documents but that differ in the details. A prototypical example is a collection of newswire articles on Latin American terrorism; each document is presumed to describe one or more terroristic acts.[1] Also defined for a given information extraction task is a *template*, which

[1]In general, documents from other domains may also be present in a collection, so that some sort of filtering must be performed either before or during extraction. For the purposes of this thesis, I consider this task ancillary to the problem of extraction and do not discuss it further.

is a case frame (or set of case frames) that is to hold the information contained in a single article. In our example, this template might have slots for the perpetrator, victim, and instrument of the terroristic act, as well as the date on which it occurred. An information extraction system designed for this problem needs to "understand" an article only enough to populate the slots in this template correctly. Most of these slots typically can be filled with fragments of text taken verbatim from the document, whence the name "information extraction."

There are many variations of the information extraction problem, which has evolved over the decade during which it has been recognized as a distinct endeavor. Some of this evolution can be traced in the proceedings of the Message Understanding Conference (MUC) (Def, 1992; 1993; 1995), the premier forum for research in conventional information extraction. In its hardest form, information extraction involves recognizing multiple entities in a document and identifying their relationship in the populated template. For example, the task might be to summarize the details of a corporate joint venture as related in a newswire article. Entities might correspond to companies; for each company the information extraction system could be required to extract descriptive information (e.g., name, nationality, 1997 net profits, etc.), as well as determine how the various companies mentioned in the article are related (e.g., "Company 3 is the joint venture of Company 1 and Company 2"). On the other end of the spectrum is the problem of generic entity recognition (or "named entity extraction"), in which the type of the entity is relevant, but not its particular role in the document. Simply finding all company names is an example of this kind of problem.

In a typical MUC problem there are multiple distinct sub-tasks, such as relevancy filtering, extraction, anaphora resolution, and template merging, which together contribute to the successful performance of the central task: mapping a document to a summary structure. Cardie describes the generic information extraction system as a pipeline in which these sub-tasks are performed sequentially on a document (Cardie, 1997). While in many domains good performance may require the handling of any or all of these tasks, the one task central to all information extraction problems is that of *extraction*: deciding which fragment from a document, if any, to put in a particular slot in the answer template. In this thesis I reserve the term *information extraction* to refer to this task.

As recent research in the information extraction community has shown, machine learning can be of service, both in performing this fragment-to-slot mapping (Kim and Moldovan, 1995; Riloff, 1996; Soderland, 1996), and in solving associated tasks (Aone and Bennett, 1996; Cardie, 1993; McCarthy and Lehnert, 1995; Riloff and Lehnert, 1994; Soderland and Lehnert, 1994). At the same time, machine learning researchers have become increasingly aware of the information extraction task as a source of interesting problems (e.g., (Califf, 1998)). This development is part of a general growth of interest on the part of the machine learning community in problems involving text, which in turn can be attributed to the growth of the Internet and the Web as a source of problem domains. In fact, some machine learning researchers, in pursuit of automated methods for handling certain Web and Internet problems related to data mining, have discovered their affinity to research done in information extraction (Doorenbos *et al.*, 1997; Kushmerick, 1997;

Soderland, 1997). Thus, a natural meeting appears to be in progress, between information extraction researchers, who have discovered the utility of machine learning methods, and machine learning researchers, who as part of an interest in data mining and textual problems have come to appreciate information extraction as a source of domains to motivate the refinement of existing machine learning methods and the development of new ones.

One consequence of this meeting is a broadening in scope of the information extraction problem. Although traditional research in information extraction has involved domains characterized by well-formed prose, many domains for which information extraction solutions would be useful, such as those consisting of Web pages and Usenet posts, do not have this character. In many cases, it may be hard to analyze the syntax of such documents with existing techniques. Grammar and good style are often sacrificed for more superficial organizational resources, such as simple labels, layout patterns, and mark-up tags. It is not the case that Web pages and Usenet posts are formless; rather, standard prose is replaced by domain-specific conventions of language and layout. Researchers have shown for particular text genres that these conventions and devices can be used to learn how to perform extraction (Califf, 1998; Doorenbos *et al.*, 1997; Kushmerick, 1997; Soderland, 1997).

1.2 Point of Departure

Several projects have investigated the possibility of performing information extraction in unconventional domains. The typical project picks a target domain and develops a learner that works well with it, but which may not be applicable to a different domain. My goal, in contrast, is a package of machine learning techniques that are applicable to as many information extraction domains as possible. Consequently, this dissertation asks the following kinds of questions:

- What level of performance is possible with domain-independent learners?
- How might such learners be structured, and what kinds of information would they use for learning?
- Can we design learning approaches to which domain-specific information can be easily added?
- Can we integrate multiple learners usefully?

These considerations determined the kind of research I conducted and the ultimate character of this thesis, which might be distinguished from similar work in three ways: in its study of diverse domains, its comparison of multiple learners, and its investigation of multistrategy learning. In the remainder of this section I elaborate each of these points in turn.

1.2.1 Study in Diverse Domains

My formalization of the information extraction problem, which is presented in Chapter 2, is sufficiently broad to cover a wide range of problems. The empirical studies presented in this thesis rely on three very different domains:

- A collection of seminar announcements posted to electronic bulletin boards at Carnegie Mellon. Announcements vary considerably in their reliance on well-formed prose, and all contain unparsable segments, such as headers. The task in this domain is to identify distinguishing details of an upcoming seminar.
- A subset of documents from the Reuters collection belonging to the *acquisition* class (Lewis, 1992). All documents in this set are written to journalistic standards. The task is to identify the parties involved in the acquisition along with other relevant details.
- Project and course pages from the World Wide Web sites of four large computer science departments. Here, the task is to find details such as the name and number of a course, or the names of a projects members and affiliates.

While other researchers have described machine learning approaches for each of the text genres on which these domains are based—Usenet posts, newswire articles, and World Wide Web pages—no study has applied a single fixed learner to such a diverse set of problems. The comparison is enlightening, clearly highlighting some of the strengths and weaknesses of the domain-independent approaches I have studied.

1.2.2 Comparison of Multiple Learners

The field of machine learning encompasses a rich variety of paradigms according to which a new algorithm might be constructed. Candidate paradigms include decision trees, relational learning, artificial neural networks, grammatical inference, instance-based learning, and statistical approaches such as Naive Bayes. Rather than restrict my attention to a single paradigm, I find it more interesting to compare learners drawn from diverse paradigms. Because the performance of any single learner on information extraction tasks is almost always substantially worse than human performance, it is hard to assess the significance of a learner's achievement by considering it in isolation.[2] Therefore, it is important to compare learners with each other, or with reasonably competent non-learning algorithmic approaches. This thesis is the first to conduct such a comparison for information extraction. It presents four learning approaches to information extraction—a "rote" learner, a statistical learner, an enhancement of the statistical learner using grammatical inference, and a relational learner—and compares them using each of the three domains described in the previous section.

[2]Incidentally, human performance is not necessarily 100%. A study conducted as part of MUC-5 rated human labelers at about 82% precision and 79% in a domain involving technical micro-electronic texts (Will, 1993). The best computer systems achieved about 57% precision and 53% recall on the same task.

	Rote	Bayes	GI	BayesGI	SRV
What does training entail?	Verbatim storage of field instances.	Construction of multiple term-frequency tables.	Construction of a regular grammar to represent the abstract structure of field instances.	Independent training of Bayes and GI.	Top-down induction of logical rules to separate field instances from background text.
What does testing entail?	Comparing fragments against learned dictionary for exact matches.	Estimating class membership from evidence provided by position, size, and individual tokens of a fragment.	Determining whether learned grammar accepts a fragment expressed in learned abstract representation.	Taking the product of estimates returned by each of Bayes and GI.	Comparing fragments against set of learned rules to see if any match.
Form of learned hypothesis	A dictionary of verbatim field instances.	A collection of frequency tables.	A stochastic regular grammar.	A Bayes classifier and a GI classifier.	A set of logical rules.
Passes through training corpus	1. Build dictionary. 2. Count true and false matches.	1. Build frequency tables. 2. Set prediction threshold.	1. One pass for each entry in transducer. 2. Transduce field instances.	Sum of Bayes and GI.	One pass for each literal in each rule.
Meaning of confidence	Specificity of matching dictionary entry.	Naive estimate (log probability) that fragment is a field instance.	Probability that learned grammar produces a fragment, given that it belongs in the language defined by the grammar.	Naive estimate (log probability) that fragment is a field instance, including language-membership estimate from GI.	Combination of estimated accuracy of all matching rules. Accuracy estimates come from rule validation on a hold-out set of documents.
Typical range of confidence	$(0, 1)$	< -20.0. Depends on problem. A score of -20.0 represents very high confidence.	$(0, 1)$. Typically closer to 0.	Approximately same as Bayes.	$(0, 1)$
Fragment context?	No.	Yes.	No.	Yes.	Yes.
Flexible context?	No.	No.	No.	No.	Yes.
Use of background text	Used only to determine specificity of dictionary entries.	Used to form estimates for tokens in immediate vicinity of fragments. In BayesIDF, background text is also used in heuristic modification of all individual token estimates.	Used in some methods for inferring transducers. Not used in induction of grammar.	See entries for Bayes and GI.	Set of "negative examples" explicitly enumerated from background text. Also available for exploration of fragment context.
Use of token features	Only literal tokens used.	Only literal tokens used.	Used in induction and application of abstract fragment representation (transducers).	See entry for GI.	Selected and applied by learner in search for best rules.
Account for whole fragment?	Yes.	Yes, and static, limited context.	Yes.	Yes.	Rules may express constraints on certain fragment tokens while ignoring others.

TABLE 1.1: Overview of the learners described in this thesis.

Table 1.1 compares the four learners described in this thesis. Note that the method called GI in the table is not really considered a standalone learner (it typically performs poorly in isolation), but is used as part of an augmentation of Bayes. It is given a separate column to make it easier to compare bayesgi, the augmented Bayes, with the other three learners. The table presents the following rows:

- **What does training entail?** What is the essential activity of the learner during induction?
- **What does testing entail?** The individual item in the database record to be filled out in response to a document is called a *field*. In Chapter 2, the extraction problem is framed as the problem of assessing field membership of individual text fragments in a document. How does each algorithm perform this assessment?
- **Form of learned hypothesis** What does a learner output after its processing of training data?
- **Passes through training corpus** In general, a learner may make an arbitrary number of passes through the training corpus. How many does a learner make, and what is the purpose of each?
- **Meaning of confidence** Each learner is designed to produce a confidence whenever it sees a fragment in a test document which it believes is an instance of the target field. What does this confidence represent?
- **Typical range of confidence** What range of values does a prediction confidence take?
- **Fragment context?** Does a learner pay attention to a fragment's context (the tokens in its immediate vicinity) in assessing its membership in the target field?
- **Flexible context?** If a learner does use fragment context, is it free to determine the size of the context to be used?
- **Use of background text** How are training tokens used which are not labeled as part of field instance fragments?
- **Use of token features** "Token features" are abstractions over individual tokens. Some learners only consider the literal tokens in building hypotheses. Does a learner use these abstractions, and, if so, how?
- **Account for whole fragment?** In assessing a test fragment, is a learner constrained to account for every part of it, or can it "choose" to examine only certain salient tokens?

Some of the entries may not make sense unless one has first read the corresponding chapters. Table 1.1 is intended, therefore, to give a quick overview of the range of approaches taken in this thesis, and to support comparisons during and after perusal of individual chapters.

1.2.3 Investigation of Multistrategy Learning

An increasingly common technique in machine learning is to combine multiple learners in an effort to realize better performance than any individual learner. *Multistrategy learning* appears especially well suited for information extraction. Such factors as the typical lack of dominance of any single approach and the rich set of possible problem representations argue in favor of hybrid or voting approaches. This thesis goes one step beyond a comparison of individual learners to ask whether there is any profit in combining them. Not only is one of the learners I study a hybrid—one that performs much better than either of its constituents—but I also present several methods for combining arbitrary learners. Experiments show that, using the best of these methods, it is almost always possible to realize better performance than that of the best constituent learner.

1.3 Claims

This thesis makes three central claims. These claims embody its emphasis on informal text and multistrategy learning.

> **Learning without Linguistics Claim** *Effective information extraction is often possible without recourse to natural language processing.*

The bulk of work on information extraction, including research in the uses of machine learning for information extraction, has assumed some kind of linguistic pre-processing. In contrast, this thesis sets out to show that, often, no such pre-processing is necessary in order for effective learning to take place.

Without doubt, some information extraction problems require, for maximal performance, that syntactic and semantic information be taken into account. There are many domains, however, where such information is either difficult to obtain—the seminar announcements and Web page domains are good examples—or simply unnecessary. Several researchers have demonstrated that in certain domains involving Web pages it is often sufficient to look for patterns involving HTML tags in order to learn perfect or nearly perfect extractors (Kushmerick, 1997; Muslea *et al.*, 1998). This thesis complements their work by developing *general-purpose* learners which by default look for directly accessible patterns in any domain to which they are applied.

Note that the acquisitions domain used in this thesis is one for which we would expect some linguistic information to be necessary for good performance. This domain serves as a good touchstone by which the limitations of the learners described in this thesis can be judged.

> **No Best Learner Claim** *There is no single best learning approach to all information extraction problems.*

This claim is impossible to prove empirically. The evidence gathered in this thesis, however, lends it strong support. Certainly, learners may vary in sophistication and flexibility, and more sophisticated learners may perform better on average, but there is too much variety in the set of typical information extraction problems for any single fixed approach to excel universally.

Papers on the topic of learning for information extraction, and in machine learning generally, often present a single algorithm, demonstrating its effectiveness on a single problem. The *No Best Learner* claim, and the evidence mustered here in support of it, is intended to argue for a comparative methodology. We will not really understand information extraction until we have identified the key problem types it encompasses and found the best methods for each type. Explaining the strengths and weaknesses of various learned extractors in a comparative setting is a step in this direction.

> **Multistrategy Learning Claim** *By combining trained information extractors we can realize substantial improvements over the performance of the best individual extractor.*

If the *No Best Learner* claim is correct, then there is good reason to believe in this claim. A multistrategy approach can always manage to perform *as well as* the best individual learner by always choosing to trust the best learner for any given problem and making that learner's predictions its own. What is more, in a setting in which all learners leave plenty of room for improvement—as is the case in many, if not most, information extraction tasks—we can hope to achieve even better performance by choosing from among learners on an example-by-example basis. I may have liked Learner One's prediction on the previous document but have reason to believe that Learner Two is better situated to make the right prediction on this document.

Of course, the question is, how do I choose whom to trust? This thesis presents two different algorithmic ways of making this decision. One way involves coupling two different learners to produce a hybrid learner. Every extraction decision by this hybrid learner, whether to accept or reject a candidate fragment of text, is the result of input from both constituent learners. The other way of making the decision takes an arbitrary number of learners and treats them as black boxes, modeling their behavior on part of the training set and using the resulting models to determine how to handle their predictions on test documents.

1.4 Thesis Organization

Before we can apply machine learning to information extraction, we must settle on a basic representation of the problem that is compatible with machine learning assumptions. Chapter 2 presents this representation in the form of a formalization. In the same chapter, the problem of evaluation is discussed, and the metrics are presented by which I judge performance throughout the thesis. To motivate my emphasis on multistrategy learning, and as an illustration of the kinds of information available in a typical document, the notion of a document *view* is introduced, and several example views are shown of a document from the seminar announcement collection.

Chapters 3, 4, and 5 are each devoted to describing various learning approaches to information extraction and presenting experimental results in several domains. Chapter 3 introduces "term-space" learning, learning which only involves term and phrase frequency statistics. Two learners are defined in this chapter, Rote, a memorizing learner, and BayesIDF, a statistical learner based on Naive Bayes as used in document classification.

Chapter 4 describes an enhancement of BayesIDF using grammatical inference. Each of the two learners, BayesIDF and an algorithm based on the grammatical inference method Alergia, is trained to perform extraction separately. A third learner is then derived by tightly coupling the predictions of the two component learners. Also discussed is the need for *transduction* as a pre-processing step to grammatical inference: In order for grammatical inference to be effective, text fragments must first be represented in terms of symbols from a small alphabet. How to transform text fragments into these symbol sequences is treated as a learning problem in its own right. The solution involves a set of simple token features.

Chapter 5 presents SRV, a relational learner. SRV uses the same kinds of features as are used as part of grammatical inference, but can apply them more flexibly. Experiments in all three thesis domains demonstrate SRV's versatility. Two case studies show how SRV's default feature set can be extended to capture genre-specific information.

Chapter 6 presents a second kind of multistrategy learning in which all learners contribute to an extraction decision for each document. Learners are treated as black boxes, and their behavior is modeled using regression and cross-validation on the training set. The resulting models are used to decide, on a document-by-document basis, which learner's prediction to make official. Three variants of this idea are presented and tested on extraction tasks from three domains. The best of these combining methods almost always yields performance improvements over the individual learner that is best on a problem.

Chapter 7 discusses related work, and Chapter 8 concludes. Appendix A presents some additional details of the domains used as part of this research, and Appendix B presents excerpts each of these domains illustrating typical patterns. Appendix C describes the tokenizing library at the heart of all learning algorithms implemented for this thesis.

Chapter 2

The Problem Space

> In this chapter I define the central problem addressed in this thesis—learning a mapping from documents to fragments—and discuss how it might be cast as a machine learning problem so that standard approaches can be brought to bear. This discussion provides a conceptual framework for the implementation of learners, but does not tell us what kinds of information to look for in a document during their design. The notion of a document *view* serves as motivation in this regard, and several important views are enumerated. Next, I present the performance criteria used to measure learner performance throughout the thesis. Finally, I discuss to what extent the task I address fits into the traditional information extraction mold, as defined by work presented at the Message Understanding Conference.

The term *information extraction* has come to refer to a number of related problems. Originally, as a discipline within NLP, it denoted a kind of directed document summarization, and was the focus of one track of the TIPSTER initiative. Over its history, however, as researchers outside the field of NLP have become interested in the problem, it has been applied to any kind of *text mining*, extracting machine-usable data from textual documents.

When I say "information extraction," I usually mean something much more precise. This chapter attempts to define the learning task rigorously. The resulting formal framework provides a nice opportunity to think about how the learning task might be approached, and motivates a discussion of document *views*. Finally, I compare the problem defined in this chapter with that addressed at the Message Understanding Conference (MUC), the premier forum for work in traditional information extraction.

2.1 Problem Definition

In most of the experiments presented in this thesis, the task to be learned amounts to the following: *Find the best unbroken fragment of text from a document that answers some domain-specific question.* If the domain consists of a collection of seminar announcements, for example, we may be interested in the location of the seminar described in a given announcement. I call the question to be answered a *field*; the fragment that answers it I call a *field instance*. Thus, one instantiation of the *location* field in our seminar announcement domain might be "`Wean 5409`" (i.e., a text fragment giving a room number). An information extraction task typically involves several fields (e.g., the location, speaker, and start time of a seminar); I regard each as a separate learning problem.

2.1.1 Formal Framework

Central to any information extraction effort is the individual document. Let D represent such a document. We can view D as a sequence of *terms*, $t_1, \cdots, t_n$, where a term is either a word or a unit of punctuation.

A *field* is a function, $\mathcal{F}(D) = \{(i_1, j_1), (i_2, j_2), \cdots\}$, mapping a document to a set of fragments from the document. The variables i_k and j_k stand for the indexes of the left and right boundary terms of fragment k. Note that some fields are not instantiated in every document (not every seminar announcement lists a location). In this case, $\mathcal{F}$ returns the empty set. I will usually assume that all fragments in $\mathcal{F}(D)$ refer to the same entity, so that it is sufficient to identify any member of $\mathcal{F}(D)$.

Given the extension of $\mathcal{F}$ for some set of training documents (i.e., given documents annotated to identify field instances), the goal of a learner is to find a function $\widehat{\mathcal{F}}$ that approximates $\mathcal{F}$ as well as possible and generalizes to novel documents.

As an alternative to approximating $\mathcal{F}$ directly, we can construct learners to model a function, $\mathcal{G}(D, i, j) = x$, which maps a document sub-sequence to a real number representing the system's confidence that a text fragment (i, j) is a field instance. Given a hypothesis in this form, implementing $\widehat{\mathcal{F}}$ may involve as little as iterating over a document, presenting $\mathcal{G}$ with fragments of appropriate size, and picking the fragment for which $\mathcal{G}$'s output is highest. In practice, we also want to use $\mathcal{G}$ to reject some fragments outright. We can accomplish this by associating a threshold with $\mathcal{G}$. A learner is said to have given *no prediction* if its output falls below this threshold for all fragments in a document.

2.1.2 Discussion

All of the learners described in the following chapters are constructed to learn a function in the form of $\mathcal{G}$. In other words, hypotheses concern fragments, rather than documents. Given a test fragment, a learner must either return *no* (reject it) or a confidence. Learning a function in the form of $\mathcal{G}$, as opposed to modeling $\mathcal{F}$ directly, has a number of advantages:

- It finesses the *search* problem inherent in $\mathcal{F}$; rather than attempt to locate a field instance, a learner is simply presented with all reasonable alternatives and chooses among them.
- $\mathcal{G}$ is closer in form than $\mathcal{F}$ to the kinds of functions implemented by conventional classification algorithms, and a number of approaches standard in related disciplines

can be brought to bear. For example, the text delimited by (i, j) can be regarded as a kind of miniature document, and a statistical document classification technique can be used to locate field instances.

- If a learner outputs a real number, it is possible to choose a threshold below which to disregard predictions, i.e., to trade recall for precision. More interesting, its reliability as a function of confidence can be *modeled*. A later chapter explores the use of such models for improved performance in a multistrategy setting.

2.2 Document Views

In the previous section, we formalized the notion of a document as a sequence of terms. While this formalization is necessary in order to define the learning task, it is only one of many ways to look at a document, one of many document *views*. A document is a "natural" object that has many different kinds of structure, some of which must be ignored in any given representation. Consequently, this thesis argues that *multiple representations are better than any single representation*.

What I am calling a *view* amounts to a category of structural information. Taking a view involves recognizing a specific kind of structure which is present only implicitly in the document. In this section, I identify some of the views I believe are relevant to the problem of information extraction.

2.2.1 The Terms View

The *terms view* regards a document as a sequence of terms, as formalized in the previous section. The bag-of-words model, which ignores ordering, is basically a weakening of this view.

It may seem counter-intuitive to refer to the canonical document representation as a view. Nevertheless, because it groups together more primitive document elements (characters), it performs the same function as more sophisticated views—it provides structure. Although two of the learning approaches described in later chapters assume this view, it is a rather impoverished document representation for the purpose of information extraction.

Figure 2.1 shows an electronic posting announcing an upcoming seminar in a university computer science department. Figure 2.2 depicts a terms view of the same document. Every whitespace-separated token is an element in the sequence that constitutes the document. Note how the structure we have removed in assuming the terms view makes the problem of identifying the seminar speaker or start time more difficult.

2.2.2 The Mark-Up View

The *mark-up view* of a document regards it as a sequence of terms interleaved with meta-terms, which provide role information about the terms. HTML contains explicit meta-terms

```
<0.21.3.95.14.12.11.ed47+@andrew.cmu.edu.0>
Type:     cmu.andrew.official.cmu-news
Topic:    ECE Seminar
Dates:    30-Mar-95
Time:     4:00 - 5:00 PM
Place:    Scaife Hall Auditorium
PostedBy: Edmund J. Delaney on 21-Mar-95 at 14:12 from andrew.cmu.edu
Abstract:


   COMPUTERIZED TESTING AND SIMULATION OF CONCRETE CONSTRUCTION

                          FARRO F. RADJY, PH.D.

                          President and Founder
                        Digital Site Systems, Inc.
                             Pittsburgh, PA

 DATE:  Thursday, March 30, 1995
 TIME:  4:00 - 5:00 P.M.
 PLACE:  Scaife Hall Auditorium
REFRESHMENTS at 3:45 P.M.

-----------------------
```

FIGURE 2.1: A seminar announcement.

```
< 0 . 21 . 3 . 95 . 14 . 12 . 11 . ed47 + @andrew . cmu . edu . 0 > type : cmu
. andrew . official . cmu - news topic : ece seminar dates : 30 - mar - 95 time :
4 : 00 - 5 : 00 pm place : scaife hall auditorium postedby : edmund j . delaney on
21 - mar - 95 at 14 : 12 from andrew . cmu . edu abstract : computerized testing and
simulation of concrete construction farro f . radjy , ph . d . president and founder
digital site systems , inc . pittsburgh , pa date : thursday , march 30 , 1995 time : 4
: 00 - 5 : 00 p . m . place : scaife hall auditorium refreshments at 3 : 45 p . m .
- - - - - - - - - - - - - - - - - - - - - - -
```

FIGURE 2.2: A seminar announcement as a sequence of literal terms.

(tags), but even ASCII contains "control" characters, such as tabs and carriage returns, the purpose of which is to partition terms. While the terms view is a uni-dimensional interpretation of a document, mark-up is multi-dimensional. Each tag represents an orthogonal dimension along which tokens can be described. A tag can be viewed as a Boolean function defined over tokens. In HTML, for example, we might define the *title* function to return *true* for any tokens occurring within the scope of a `<title>` tag. Still another way to regard mark-up is as the instantiation of a number of relations; the tokens occurring together in a title field all participate in a relation that distinguishes them from other tokens in the document.

Figure 2.3 shows a World Wide Web home page, and Figure 2.4 depicts a mark-up view of the same page. In this figure, all non-whitespace characters not belonging to some mark-

```
<html>
<head>
<title>Dayne Freitag's Home Page</title>
</head>

<body bgcolor="#FFFFFF">

<center><h2>Dayne Freitag</h2>
<hr>
<h3><font face="Helvetica">Contents</font></h3></center>

<center>

<table>

<tr><td>
<font face="Courier">
Introduction....................................
<a href="intro.html"><i>intro.html</i></a>
</font>
```

FIGURE 2.3: Part of a personal home page from the World Wide Web.

```
<html>
<head>
<title>***** ********* **** ****</title>
</head>

<body bgcolor="#FFFFFF">

<center><h2>***** *******</h2>
<hr>
<h3><font face="Helvetica">********</font></h3></center>

<center>

<table>

<tr><td>
<font face="Courier">
************************************************
<a href="intro.html"><i>**********</i></a>
</font>
```

FIGURE 2.4: A mark-up view of the excerpt shown in Figure 2.3, in which non-markup, non-whitespace characters have been replaced by asterisks.

```
*******************************************
*****     ****************************
******    *** *******
******    *********
*****     **** * **** **
******    ****** **** **********
********* ****** ** ******* ** ********* ** ***** **** **************
*********

   ************ ******* *** ********** ** ******** ************

                          ***** ** ****** *****

                          ********* *** *******
                        ******* **** ******** ****
                             *********** **

 *****  ********* ***** *** ****
 *****  **** * **** ****
 ******  ****** **** **********
************ ** **** ****

***********************
```

FIGURE 2.5: A layout view of the document shown in Figure 2.1, in which non-whitespace characters have been replaced by asterisks.

up element have been replaced by asterisks. With a little experience with similar pages, a human can identify the name of the home page's owner, with reasonable confidence, from the mark-up view alone.

2.2.3 The Layout View

The *layout view* of a document regards it as a two-dimensional arrangement and sizing of terms. This view can be regarded as an interpretation of the mark-up view by some application.

In general, many important textual objects can be discerned only at this level, such as paragraphs, headlines, tables, mail headers, signatures, etc. Such objects are frequently employed to separate a field from surrounding text, or to associate it with surrounding text in a special way. Textual tables, for example, imply something about the text fragments they comprise; columns typically define an attribute over multiple objects, while rows associate the various attributes of a single object.

Figure 2.5 depicts a layout view of the document shown in Figure 2.1. In this figure, all non-whitespace characters have been replaced by asterisks. While this view typically does not provide enough information by itself to identify field instances, it is nevertheless a useful source of information. Suppose, for example, we wanted to know for what parts of the document shown in Figure 2.5 a traditional linguistic analysis would be feasible. An experienced eye can quickly identify regions where this should *not* be attempted, such as

```
<9.99.9.99.99.99.99.aa99+@aaaaaa.aaa.aaa.9>
Aaaa:     aaa.aaaaaaa.aaaaaaaa.aaa-aaaa
Aaaaa:    AAA Aaaaaaa
Aaaaa:    99-Aaa-99
Aaaa:     9:99 - 9:99 AA
Aaaaa:    Aaaaaa Aaaa Aaaaaaaaaa
AaaaaaAa: Aaaaaa A. Aaaaaaa aa 99-Aaa-99 aa 99:99 aaaa aaaaaa.aaa.aaa
Aaaaaaaa:


   AAAAAAAAAAAA AAAAAAA AAA AAAAAAAAAA AA AAAAAAAA AAAAAAAAAAA

                          AAAAA A. AAAAA, AA.A.

                          Aaaaaaaaa aaa Aaaaaaa
                        Aaaaaaa Aaaa Aaaaaaa, Aaa.
                             Aaaaaaaaaa, AA

 AAAA:  Aaaaaaaa, Aaaaa 99, 9999
 AAAA:  9:99 - 9:99 A.A.
 AAAAA:  Aaaaaa Aaaa Aaaaaaaaaa
AAAAAAAAAAAA aa 9:99 A.A.

-----------------------
```

FIGURE 2.6: A layout view augmented with typographic information.

the mail header and centered text. What is more, a little more experience with documents from this domain should make it possible to make good approximate guesses about the location of field instances, such as the seminar's speaker.

2.2.4 The Typographic View

The *typographic view* amounts to a collection of simple functions defined over the tokens in a document. These functions reflect membership of the characters constituting a token in a number of character classes. These classes, such as *numeric*, *punctuation*, and *upper-case*, do not serve to contain meaning so much as organize the text in a way that makes it more readily digestible. This view is a powerful source of information for certain information extraction problems. Because of this, and because it is easy to analyze text for typographic information, two learners we will describe in later chapters make extensive use of this view.

Figure 2.6 shows the layout view augmented with typographic information. In this figure, punctuation has been passed through unaltered, numeric characters have been replaced with `9`, lower-case alphabetic characters with `a`, and upper-case characters with `A`. To a trained eye, it becomes possible in this view to locate instances of most of the fields with high reliability.

```
        +--------------A-------------+       +---------Jp--------+
        |                            |---Mp--+      +------A-----+
        |                            |       |      |            |
computerized.a [testing] [and] simulation.n of concrete.a construction.n
```

FIGURE 2.7: A syntactic view of the title of the seminar announced in Figure 2.1.

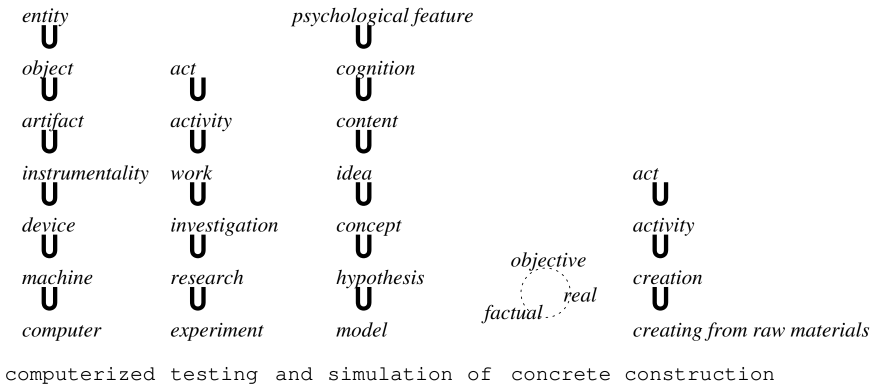

FIGURE 2.8: A semantics view, produced using Wordnet, of the title of the seminar announced in Figure 2.1.

2.2.5 The Linguistic View

A linguistic view of a document regards it as a syntactically and semantically structured object. Document terms participate in a set of syntactic relations that bind them together in graphical structures. Each content word possesses one or more semantic senses, only one of which is in effect in a given context.

Our understanding of how linguistic structure is recovered from a document is incomplete, as many open questions in NLP serve to demonstrate. This, combined with the inherently multi-dimensional nature of linguistic structure, makes it hard to depict a linguistic view in graphical form. Figures 2.7 and 2.8 attempt to present the syntactic and semantic structure, respectively, of the seminar title from Figure 2.1. In Figure 2.7 syntax consists of a set of binary relations or "links," as produced using the link grammar parser (Sleator and Temperley, 1993). Each relation binds together terms in a sentence according to a particular syntactic role. In Figure 2.8 the $\cup$ symbol stands for the *is-a* relation, which in Wordnet is restricted to nouns and verbs. Adjectives and adverbs, in contrast, are organized into clusters of related meanings. Note that none of the text in this seminar announcement is perfectly suited for linguistic processing, since it contains no complete sentences and resorts to non-linguistic devices to relay information. Thus, it argues eloquently for the use of some of the other views when performing information extraction in such domains.

Clearly, however, the linguistic view is the most powerful source of information for extraction in many cases—if we can make effective use of it. Unfortunately, the very existence of information extraction as a discipline implies shortcomings in current NLP methods.

2.3 Evaluating Performance

The problem of evaluating the performance of an information extraction system is surprisingly subtle. Here, I outline the space of possible performance measures and define the metrics I use in this thesis.

2.3.1 Unit of Performance

For most problems I study, all field instances in a document D—all members of the set $\mathcal{F}(D)$—refer to the same underlying entity. In a seminar announcement, for example, the start time, which is unique for a given seminar, may be listed several times. I call this the *one-per-document* (OPD) setting. The unit of performance for OPD problems is the individual document. The central question is: *In looking for Field X, did Learner Y act appropriately on Document Z?* This question is posed for each Document Z in the test set. For each document, for which the answer to this question is *yes*, the learner is credited with one correct response.

In a few cases, I study problems for which we expect each field instance in a document to represent a distinct underlying entity. For example, Web pages describing research projects often list project members; if the object is to extract member names, then it is inappropriate just to take a learner's top prediction. I call this the *many-per-document* (MPD) setting. For MPD problems, which are typically harder, I pose a different question: *Did Prediction X made by Learner Y identify an instance of Field Z?* In other words, the unit of performance for these problems is the individual prediction.

In the discussion that follows, and through most of the thesis, I assume the OPD setting. Most of the considerations I bring up, however, apply to the MPD setting, as well.

2.3.2 Document Outcomes

Given a document and a set of predictions (fragment boundaries) from a learner, we can take the learner's most confident prediction as its official estimate. Call this prediction P and let $P = \text{nil}$ represent non-prediction. There are four possible outcomes:

Correct A field instance is correctly identified ($P \in \mathcal{F}(D)$).

Wrong The best prediction is not a field instance ($P \notin \mathcal{F}(D) \wedge \mathcal{F}(D) \neq \emptyset$).

Spurious The system predicts for a document in which the field is not instantiated ($P \neq \text{nil} \wedge \mathcal{F}(D) = \emptyset$).

No prediction The system makes no prediction ($P = \text{nil}$).

The learner's *precision* is defined as:

$$Precision \equiv \frac{correct}{correct + wrong + spurious}$$

In other words, precision is the fraction of correct document outcomes divided by the number of all files for which the system makes some prediction.

Note that this is not the only reasonable performance metric. There are at least two variations which make sense, depending on the criteria imposed by the application we have in mind. For example, if we are mainly interested in ensuring that when a field is instantiated it is retrieved (i.e., we emphasize *recall*), then we may choose not to count spurious predictions as errors. This amounts to treating documents in which a field is not instantiated as irrelevant.

We may also want to count as errors those cases for which an extraction was possible but the learner made no prediction, as is commonly done in the MUC evaluations. I do not do this for two related reasons:

- I assume that the learning approaches studied in this thesis will serve as components of a larger system. A learner may issue predictions on only a small fraction of documents but with high reliability. If we count its failure to predict as errors, we are obscuring its usefulness to a larger system that treats the learner's predictions as one among many sources of information.

- I want to investigate precision/recall behaviors (see below). This requires that non-predictions (or predictions below some threshold) be treated as irrelevant in measuring precision.

2.3.3 Fragment Outcomes

The document outcome depends on a learner's most confident prediction, which takes the form of a fragment from the document. Thus, how we determine whether this prediction is correct is important. There are three basic criteria we might use:

Exact The predicted instance matches exactly an actual instance.

Contain The predicted instance strictly contains an actual instance, and at most k neighboring tokens.

Overlap The predicted instance overlaps an actual instance.

Each of these criteria can be useful, depending on the situation, and it can be illuminating to observe how performance varies with changing criteria. The *overlap* criterion shows how good a method is at approximately identifying the location of instances, without penalizing for misidentified boundaries. The *contain* criterion is potentially useful for showing how well an approach will serve in some applications, especially those involving end users, who can easily filter out erroneous text included from an instance's context. Some learners show considerable variability under changing criteria, while others do not.

Of course, the only criterion that is useful for all possible applications is the *exact* criterion. When the criterion is not explicitly stated, performance numbers assume this criterion.

2.3.4 Precision and Recall

The precision metric defined above does not count documents for which a learner offers no prediction. An alternative metric might count as errors those documents in which a field is instantiated for which a learner makes no prediction. This metric would have the effect of making a reticent learner look bad, when in fact it might be performing quite well on a subset of the testing documents. Instead, I account for this failure to predict by means of the complementary metric (which, like precision, is also standard in information retrieval):

$$\mathit{Recall} \equiv \frac{\mathit{correct}}{|\{D|\mathcal{F}(D) \neq \emptyset\}|}$$

or the number of correct predictions divided by the total number of documents that contain at least one field instance. Wherever a precision number is given, a corresponding recall number will be included. It is important to consider both numbers when comparing results.

A learner responding to only a few documents typically chooses those documents for which its bias is best suited, those documents that are "easiest" for it. Such a learner will generally achieve better precision than a learner that offers predictions for all documents, easy and hard. In practice, therefore, there is often an inverse relation between recall and precision. Measures taken to afford higher recall often result in lower precision, and vice versa.

Although the notion of not predicting is built into some learners (e.g., a rule learner may have no rules that match a particular document), so that they naturally do not issue a prediction for every document, it is always possible to make a learner *less* responsive by raising its confidence threshold. Consequently, it is possible to turn a high-recall learner into a low-recall one, either for the purpose of achieving a desired precision level, or of comparing it with other learners at a given recall level.

A *precision/recall graph* depicts the effect of manipulating the confidence threshold in this way. Figure 2.9 shows one such graph. To generate the graph for a learner, all of its predictions (each prediction that was highest for some test document) are sorted in non-increasing order by confidence. Each point on the horizontal axis corresponds to some fraction of these predictions. For example, 0.5 on this axis represents the top 50% of predictions, according to confidence. The vertical value at this point shows the precision of these predictions. By looking at such a graph, we may see that a learner with mediocre precision at nearly 100% recall actually performs perfectly at 90% recall. To judge such a learner based on its full-recall performance would be a mistake, since by throwing out

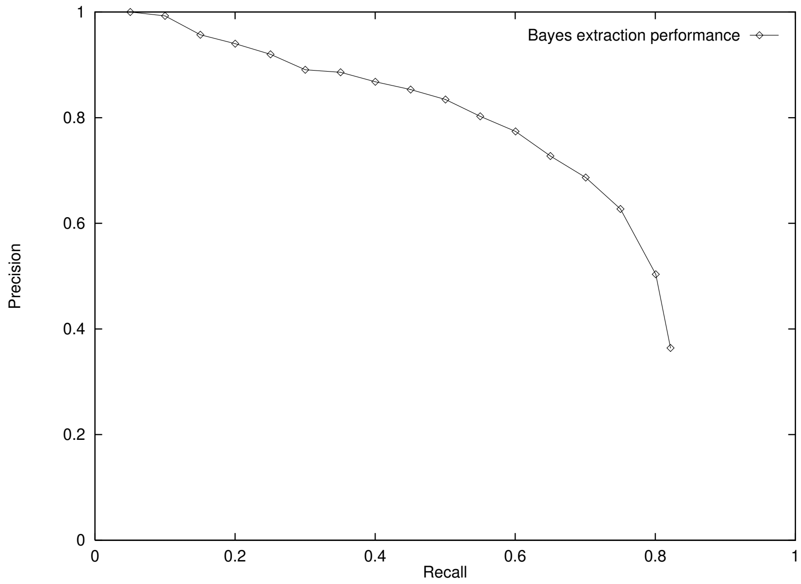

FIGURE 2.9: A precision/recall graph.

a small fraction of its correct predictions we can put it to good use. In general, as in Figure 2.9, we expect to see a more or less smooth decline in precision with increased recall. How well a particular plot fits this expectation tells us something about how well a learner's confidence correlates with the reliability of its predictions.

In lieu of precision/recall graphs, I sometimes present results in the form of tables listing performance at various fixed recall levels. Even though such tables do not give a comprehensive picture of a learner's precision/recall behavior, they at least allow us to examine a learner's precision on those documents for which it is most confident. Another advantage of such tables is that they make it convenient to use error margins. In the typical machine learning setting, a single measure—usually accuracy—is used to compare two or more learners. This number is typically the result of N classification attempts, where N is fixed for all learners. It is standard practice, in such a case, to present error margins, so the significance of a learner's achievement can be assessed. Here, in contrast, we are using two inter-dependent metrics—precision and recall—and the inclusion of error margins appears less useful. Just because one learner reaches higher precision than another does not necessarily make it better, because the corresponding recall number may be much lower. In such a case it does not help to know that the learner's better precision is statistically significant. In contrast, if two learners are compared at a fixed recall level, then a single statistic decides the outcome, and error margins make more sense. My practice, therefore, is to present error margins only when comparing learners at a fixed recall level.[1] All error margins in this thesis represent 95% confidence.

[1]Note that, even though the recall level is fixed, the comparison is not based on the same number of tests—unless the learners also achieve the same precision!

For the purpose of comparing learners, it can be awkward to examine two performance numbers for each learner. The F-measure (van Rijsbergen, 1979), as used in information retrieval, provides a method for combining precision (P) and recall (R) into a single summary number:

$$F = \frac{(\beta^2 + 1.0)PR}{(\beta^2 P) + R}$$

The parameter β determines how much to favor recall over precision. It is typical for researchers in information extraction to report the F1 score of a system ($\beta = 1$), which weights precision and recall equally. Although it can be difficult to say what an F1 score represents in operational terms, the single performance number allows a convenient comparison of information extraction systems. I find it most illuminating to present the *peak F1*, the F1 score of that point on the precision/recall curve for which it is maximized. The F1 score of the rightmost point in Figure 2.9 is about 50 (36% precision at about 82% recall). The best F1 score on this curve, however, is at 70% recall, where an F1 score of 69 is reached (69% precision). Because this learner is optimized for recall, it makes many spurious low-confidence predictions. To use its point of highest recall in a presentation of F1 scores would obscure its strengths.

To compute peak F1, I the precision/recall curve of a learner is sampled at 1% intervals. At each such point, the F1 score is computed. The highest of this collection of F1 scores is presented. As with precision, it is interesting to ask when there is a clear winner among several competing methods. Throughout this thesis, therefore, in tables comparing the peak F1 scores returned by multiple learners, if a score is presented in bold face, it is the best score in statistical terms— the single best score, such that its improvement over the next best score is judged statistically significant with 95% confidence. The scores shown in this thesis are always the result of multiple independent runs, and in each such run the training and testing sets are the same for all learners. To make the judgment of statistical separation, therefore, a "paired t-test" is used. For each run, the peak F1 score is determined for the best learner (Learner A) and next-best learner (Learner B)—as shown by the complete averaged results—and the difference between the two scores calculated. If a t-test over these differences supports the hypothesis that the difference in peak F1 between Learner A and Learner B is greater than 0 with 95% confidence, then Learner A's score appears in bold face.

2.3.5 Problem Difficulty

Information extraction is a challenging problem. On many of the individual extraction tasks described in the thesis, precision and recall of even the best learners is well below 1.0, but in interpreting such results it is important to keep in mind what learners are being asked to do. Confronted with a sequence of tokens, a learner must select a sub-sequence. In most cases, if the fragment it selects differs from the "right" answer in any way—if, for example, it includes one token too many—its selection is counted as an error. Often, depending on the task, there is only a single fragment that is considered correct; usually, there are five or fewer. Thus, precision and recall results that are less than perfect must be weighed against the large number of fragments that might be selected inappropriately.

Information retrieval provides a measure—*fallout*–of the degree to which a system's performance is degraded by the availability of a large number of irrelevant documents. If *Irrelevant* is the total number of irrelevant documents, and *FalsePos* is the number of these which a system inappropriately labels relevant, then

$$Fallout \equiv FalsePos/Irrelevant$$

measures the tendency of a system to be led astray by irrelevant documents. If we say that field instances are relevant objects, and all other fragments of appropriate size—any fragment containing no fewer tokens than the smallest training instance, and no more tokens than the largest—constitute the set of irrelevant objects, then we have one measure of the degree to which a system successfully copes with the inherent difficulty of the extraction problem.

By this measure all learners described in this thesis do quite well. Even at maximum recall, when precision is lowest, no learner suffers more than 1% fallout. Because fallout numbers are consistently so small, and because in a comparison of learners fallout does not lead to conclusions any different than those supported by precision, I do not present fallout as part of the experimental results. I mention it here in order to place less-than-perfect precision/recall results in persective.

Another way to appreciate the difficulty of an extraction task is to measure the performance of a strawman algorithm. For each task, Appendix A shows, among other things, the performance of an algorithm that issues random guesses. The guessing game is strongly biased in the strawman's favor in the following way: For each test document, the strawman is "told" how many field instances it contains, and for each such instance, it is allowed to select, at random, some fragment of the same length. On one-instance-per-document tasks (all but two of the tasks studied here), if *any* of the strawman's selections matches a field instance, its performance is counted as correct. Its performance is then measured according to the same criteria as that used for the other algorithms.[2]

Strawman accuracy ranges from about 0.5% to almost 8%, depending on the task, but for most problems it is close to 1%. The problems on which the strawman scores much higher than 1% are those in which documents tend to contain many instances of a field; because the strawman is allowed to issue one prediction for each instance, the likelihood that any of its predictions will match a field instance is higher with such documents. Notwithstanding the favorable circumstances under which the strawman is tested, the performance of even the least successful learner is well above that of the strawman on most problems. It is clear, therefore, that learning, in whatever form, is making substantial inroads into some difficult problems.

[2]Note that because it always issues exactly the same number of predictions as there are field instances in a test document, its precision, recall, and F1 are always the same number—what Appendix A calls *Accuracy* to avoid confusion.

2.4 Domains

Three document collections and four information extraction problem domains formed the basis of the individual extraction tasks addressed in this thesis. The three collections from which documents were drawn differ widely in terms of the purpose and structuredness of individual documents.

The seminar announcement collection consists of 485 electronic bulletin board postings distributed in the local environment at Carnegie Mellon University. The purpose of each document in this collection is to announce or relate details of an upcoming talk or seminar. Announcements follow no prescribed pattern; documents are free-form Usenet-style postings. I annotated these documents for four fields: *speaker*, the name of a seminar's speaker; *location*, the location (i.e., room and number) of the seminar; *stime*, the start time; and *etime*, the end time.

The acquisitions collection consists of 600 documents belonging to the "acquisition" class in the Reuters corpus (Lewis, 1992). These are newswire articles that describe a corporate merger or acquisition at some stage of completion. I defined a total of ten fields for this collection:

- **acquired** the official name of the company or a short description of the resource in the process of being acquired
- **purchaser** the official name of the purchaser
- **seller** the official name of the seller
- **acqabr** the short form of *acquired*, as typically used in the body of the article (e.g., "IBM" for "International Business Machines Inc")
- **purchabr** the short form of the purchaser
- **sellerabr** the short form of the seller
- **dlramt** the amount paid for the acquisition
- **status** a short phrase indicating the status of negotiations
- **acqloc** the geographical location of *acquired*
- **acqbus** the business of *acquired* (e.g., "bank" or "software for home entertainment")

The university Web page collection is a sample of pages from a large collection of university pages assembled by the World Wide Knowledge Base project (WebKB) (Craven *et al.*, 1998). As part of an effort to classify Web pages automatically, WebKB manually assigned each of several thousand pages downloaded from four major university computer science departments to one of six classes: student, faculty, course, research project, departmental home page, and "other." From this collection I created two sub-domains, one consisting of 101 course pages, the other of 96 research project pages. The course pages were tagged for three fields: *crsNumber*, the official number of a course, as assigned by the university (e.g., "CS 534"); *crsTitle*, the official title of the course; and *crsInst*, the names of course instructors and teaching assistants. The project pages were tagged for two fields: *projTitle*, the title of the research project; and *projMember*, the names of the project's members and alumni.

Additional details on these three collections can be found in Appendix A. Excerpts from sample documents are available in Appendix B.

2.5 MUC

As noted, I use the term "information extraction" in a more restricted sense than usual. As a discipline, information extraction is as old as the Message Understanding Conference (MUC) (Def, 1995), the forum that defined the problem and until recently set the research agenda for it. Lately, however, the idea of information extraction, as a generic term to cover all sorts of text mining, has awakened interest in the machine learning community. To avoid confusion, therefore, I will sketch the problem as it is understood by the MUC community and point to the salient differences in definition and evaluation between MUC-style information extraction and my work.

The essential components of a MUC-style information extraction problem are a collection of prose documents from some semantically coherent domain and a set of templates which define how documents are to be summarized. A MUC template is a kind of skeletal summary, providing the structure but omitting the details, which are to be found in the individual document. The simplest kind of template is a relational record schema; each item in an instantiated record is a text fragment from the corresponding document. In our seminar announcement example, we might have a template with slots for the seminar title, speaker, location, start time, and end time.

How documents are to be summarized is a question of domain definition. While a single relational template like this is adequate to convey most of the essential information in many domains, it almost always excludes some information of potential interest. Thus, MUC templates, especially those from later conferences, tend to have more complicated structure. Templates may be nested (i.e., the slot of a template may take another template as its value), or there may be several templates from which to choose, depending on the type of document encountered. In addition, MUC domains include irrelevant documents which a correctly behaving extraction system must discard. A template slot may be filled with a lower-level template, a set of strings from the text, a single string, or an arbitrary categorical value that depends on the text in some way (a so-called "set fill").

In cases where elaborate representations (nested templates, set fills) are required of a system, task difficulty may approach that of full natural language understanding. In general, the challenges facing natural language understanding cannot be circumvented in information extraction. Some semantic information and discourse-level analysis is typically required. To this are also added sub-problems unique to information extraction, such as slot filling and template merging.

FIGURE 2.10: The generic information extraction processing pipeline, according to Cardie.

Figure 2.10 shows Cardie's conception of the flow of control in a generic information extraction system (Cardie, 1997). A document is initially decomposed into a sequence of terms or tokens, which are subjected to syntactic and superficial semantic analysis. From this analysis, "Sentence Analysis" generates some representation of sentential structures. It is typical of many information extraction systems that these structures produce sentence fragments, rather than complete sentences. Many information extraction projects have found full sentence parses inefficient, noisy, and unnecessary for most information extraction problems.

"Extraction" is the process of looking through these sentential structures for text to fill constituent slots in the answer templates. When MUC researchers speak of using machine learning to perform information extraction, they usually are referring to this task. Generally, extraction produces multiple sub-templates, some of which share the same underlying entities, so must be combined to generate the correct answer template. The process of determining which of these sub-templates co-refer and how they should be combined is "Merging." Finally, "Template Generation" assembles intermediate results into the official form stipulated in the task definition.

The problem addressed here relates to MUC in the following ways:

- This thesis is concerned with methods that work in non-traditional domains, with documents consisting of grammatically ill-formed text, such as Usenet posts and Web pages. MUC documents consist of well-formed prose, and linguistic analysis is assumed to be necessary.

- Each field in a template is considered in isolation. The MUC setting groups fields into templates for both definition and evaluation. As noted, however, this special focus on individual fields (slots) is also often called "information extraction."

- This thesis does not attempt to address any of the auxiliary tasks, such as relevance detection and discourse analysis.

- In most cases, only only those problems are studied in which all field instances in a document refer to the same underlying entity. In general, of course, a field may have multiple, semantically distinct instantiations in a file. For example, a Web page describing a computer science research project at a university usually lists the names of all members (i.e., all instances of the project-member field).

The performance measures I discuss in this chapter reflect some of these commitments. In contrast with MUC, my metrics treat the individual document as the unit of performance.

In MUC, performance is measured in terms of individual slots in the key templates. A single correct answer corresponds to a response slot of the right type being filled with the appropriate text. Performance is then measured as an average over all slots, of any type, contained in the key.

How key and response templates are aligned, and what constitutes a single correct extraction, is a matter for complicated scoring software. Half credit is awarded for slots which match "partially." Otherwise, both unfilled slots and spurious slot fills are counted as errors.

The MUC scoring regime is unsuitable for the experiments described in this thesis for several reasons:

- Aligning templates and accounting for partial matches in MUC fashion is domain specific. The exact methods used at MUC are not described in published proceedings.
- Averaging performance over a diverse set of fields (slots), rather than profiling the performance on each field individually, obscures the sort of information needed to understand the behavior of individual learners.
- The MUC scoring scheme is too complicated to permit an investigation of precision/recall behavior.

Chapter 3

Term-Space Learning for Information Extraction

In *term-space* learning a document is regarded, in a case-insensitive fashion, as a sequence of terms. All other information—typography, layout, linguistics—is ignored. This chapter describes two term-space learners, Rote and Bayes. Rote memorizes field instances verbatim and only issues predictions when test fragments match previously seen instances verbatim. Bayes uses term-frequency statistics to estimate the likelihood that a novel fragment is a field instance. Experiments with the seminar announcements compare the two learners. Rote shows very good precision in identifying instances of the *location* field while achieving surprisingly high recall. The performance of one variant of Bayeson *stime* and *etime* is close to perfect. Both learners fare worse on the *speaker* field, which is characterized by uncommon tokens and less stereotypic context. Additional experiments with the acquisitions articles show that this is a more difficult domain. Nevertheless, the term-space learners show good performance on a few of the acquisitions fields.

In this chapter, I consider learning approaches that take only the terms view of a document—term-space learners. A *term* is as defined in the previous chapter: an uninterrupted sequence of alpha-numeric characters *or* a single punctuation character. Any learner that assumes no information beyond that available in such a representation, I call a *term-space* learner.

Term-space learners dispense with much of the information available in a document. This has a number of advantages:

- **It results in less domain dependence.** Because term-space learners make minimal assumptions, they are applicable to the widest variety of information extraction problems. Consider, in contrast, an approach that requires a syntactic pre-processing step in order to function. The applicability of such an approach is undermined when documents do not consist of well-formed sentences, as with many Web pages.

- **It is very efficient.** Term-space learners require less processing than approaches that seek to exploit more of the information in document. As a result, they finish much more quickly.
- **It sometimes yields the best performance.** Sometimes the benefits brought by enriching a representation or using a sophisticated learner are outweighed by the liabilities of a larger hypothesis space. Some recent work in machine learning has shown that simple learners are at least competitive and sometimes better than more sophisticated ones (Holte, 1993; Domingos and Pazzani, 1996).
- **It provides useful information.** Even if term-space approaches are not the best at a given task, they provide valuable information. All the approaches described in this thesis are intended as components in a larger information extraction system. Chapter 6 shows how term-space learners can contribute to better overall performance, even if their performance is not best on a given task.

This chapter presents two term-space learners, Rote and BayesIDF. Experiments with the seminar announcement and acquisition domains provide clear evidence that term-space approaches are useful.

3.1 Rote Learning

Perhaps the simplest possible learning approach to the information extraction problem is to memorize field instances verbatim. Presented with a novel document, this approach simply matches text fragments against its "learned" dictionary, saying "field instance" to any matching fragments and rejecting all others.

More generally, we can estimate the probability that the matched fragment is indeed a field instance. The dictionary learner I experiment with here, which I call Rote, does exactly this. For each text fragment in its dictionary, Rote counts the number of times it appears as a field instance (p) and the number of times it occurs overall (n). Its confidence in a prediction is then the value p/n, smoothed for overall frequency. Rote uses Laplace estimates under the assumption that this is a two-class problem, *field instance* and *non-field instance*. Thus, the actual confidence Rote assigns to a prediction is $(p+1)/(n+2)$.

Rote's dictionary is constructed in two passes through the training corpus. In the first pass, the dictionary, which is initially empty, is populated with field instances. At the end of this pass, the dictionary contains all distinct instantiations of a field. In the second pass, all text in the training corpus, field and non-field, is scanned in search of fragments matching dictionary entries. Whenever such a match is found, two counts associated with its dictionary entry—the p and n mentioned in the previous paragraph—are updated. The n count is always incremented in such an event; the p count is incremented only if the fragment is tagged as a field instance.

The dictionary, which is effectively a set, can be implemented using any data structure that can represent a set, such as a linear list or a hash table. Rote uses a discrimination

Fragment: `<location>Wean Hall 4601</location>`

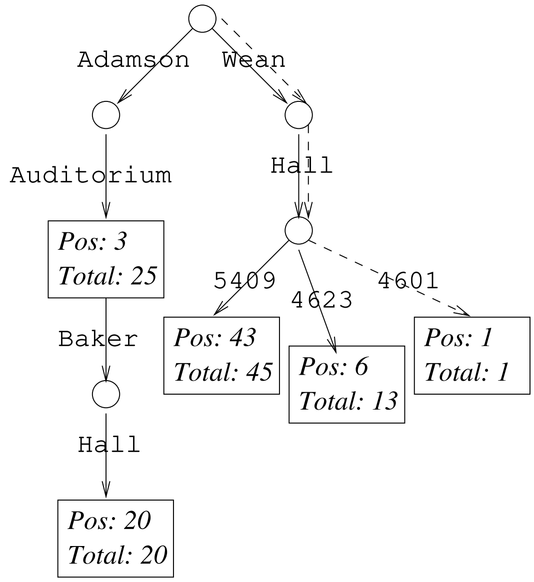

FIGURE 3.1: A hypothetical insertion of a seminar location instance into the discrimination net used to implement Rote's dictionary.

```
Function Match(tree, index)
   Return MatchInternal(tree, index, nil)
End Function

Function MatchInternal(tree, index, result)
   token = TokenAt(index)
   If Null(token)     /* Index is out of bounds */
      Return result
   End If
   node = FollowBranch(tree, token)
   If Null(node)      /* No branch found */
      Return result
   Else If Terminal(node) And Better(node, result)
      result = node
   End If
   Return MatchInternal(node, index + 1, result)
End Function
```

TABLE 3.1: Procedure for finding an entry in Rote's dictionary.

net, which is particularly suited to the multi-token nature of most field instances: Using this representation it is possible to halt consideration of a non-matching fragment when the first non-matching token is encountered. Figure 3.1 depicts an insertion into this net. Rectangular boxes in the figure represent terminal nodes, while circles represent non-terminal nodes. The dashed arrows show where the insertion is made for the phrase listed at the top of the figure.[1] Table 3.1 presents a pseudocode version of the matching procedure that is used both in the second training pass and during testing. The variable `tree` holds a pointer to the discrimination net that implements Rote's dictionary; any entry returned will match a fragment beginning at the token indicated by the variable `index`. If no matching entry is found, this procedure returns *nil*. Given multiple matching entries, it returns the best one, as determined by the function `Better()`. This function uses the statistics stored in the terminal nodes (as in Figure 3.1) to decide which of two nodes is better.

As simple as Rote is, it nevertheless is surprisingly applicable in a wide variety of domains. Of course, its applicability depends on the nature of the task, but at the very least, a Rote prediction, especially a high-confidence one, is a valuable piece of information. Because of the simplicity of the assumptions that go into a prediction, prediction confidence correlates well with actual probability of correctness.

One problem with Rote, of course, is that it cannot generalize to recognize novel instances. Applying a rote learner to the problem of document filtering, say, would involve admitting a document as *relevant* only if a user had previously identified it as such. Of course, the nature of a typical information extraction task, and the fact that field instances are generally much shorter than full documents, makes rote learning a reasonable idea.

Another problem with Rote is its insensitivity to context. The context in which a field instance appears presumably supplies *some* useful information. It is hard to imagine how Rote might be extended to exploit context in an effective way. We could elaborate the structure stored in Rote's dictionary to include k context tokens from either side of a field instance, but this would tend to counteract any benefit we realize through the statistics collected in the second training pass. Any variability in the context would result in our storing multiple entries where we would only store one in the context-insensitive version of Rote. The counts associated with these entries would be smaller and statistically less trustworthy that those associated with the corresponding context-insensitive entry.

3.2 Naive Bayes

In contrast with Rote, which must match a fragment verbatim in order to make a prediction, Bayes sums evidence provided by all tokens individually, including the tokens in a fragment's context. This section derives Bayes from Bayes' Rule and presents two modifications that appear to improve its performance.

Rather than attempt to match complete phrases, it might make sense to treat each token in and around a candidate fragment as a separate source of information, and to make a

[1] Note that a "terminal" node also can have children, since it is possible for one field instance to be the prefix of some separately occurring field instance (e.g., "5409 Wean" and "5409 Wean Hall").

statistical estimate that combines multiple individual estimates. This would overcome the limitations ascribed to Rote above:

- As long as previously unseen field instances share some of the same tokens as instances already observed, there is some possibility that a statistical approach will still recognize them. The previously unseen fragment "`Wean 7220`" might be identified as a seminar location based solely on the strength of the association of the word "`Wean`" with seminar locations.
- Incorporating estimates for tokens occurring in the text surrounding a fragment presents no difficulty. Contextual tokens contribute evidence in the same way tokens that are part of a fragment do.

And there is ample precedent for such an approach in disciplines related to information extraction. In the discipline of document classification, for example so-called bag-of-words algorithms, which include Rocchio with TFIDF term weighting (Rocchio, 1971) and Naive Bayes (Lewis and Gale, 1994), are state of the art. The algorithm I will call Bayes is in fact adapted from Naive Bayes as used for document classification.

3.2.1 Fragments as Hypotheses

Bayes' Rule tells us how to update a hypothesis H in response to the evidence contained in some empirically obtained data D:

$$\Pr(H|D) = \frac{\Pr(D|H)\Pr(H)}{\Pr(D)}$$

In other words, the posterior probability that H is correct is proportional to the product of the prior probability $\Pr(H)$ and the probability of observing the data D, conditioned on H, $\Pr(D|H)$. In classification, where the object is to choose one of several competing hypotheses H_i, the denominator $\Pr(D)$ is the same for all H_i and is typically disregarded; the hypothesis H_i that maximizes the product $\Pr(D|H_i)\Pr(H_i)$ is chosen as the best classification according to Bayes' Rule. In order to apply Bayes' Rule to classification, therefore, two estimates are needed: $\Pr(D|H_i)$ and $\Pr(H_i)$, the conditional data likelihood and the prior.

Consider now the problem of identifying the name of the speaker in a seminar announcement. We can model this problem as a collection of competing hypotheses, where each hypothesis represents our belief that a particular fragment gives the speaker's name. In this case a hypothesis takes the form, "the text fragment starting at token position p and consisting of k tokens is the speaker." (Call this hypothesis $H_{p,k}$.) In the case of a seminar announcement file, for example, $H_{309,2}$ might represent our expectation that a speaker field consists of the 2 tokens starting with the 309th token.

Given a document, a learner based on Bayes' Rule is confronted with a large number of competing hypotheses $H_{i,k}$, one of which it will choose as most probable—whichever maximizes the product $\Pr(D|H_{p,k})\Pr(H_{p,k})$. What, then, corresponds to the terms $\Pr(D|H_{p,k})$

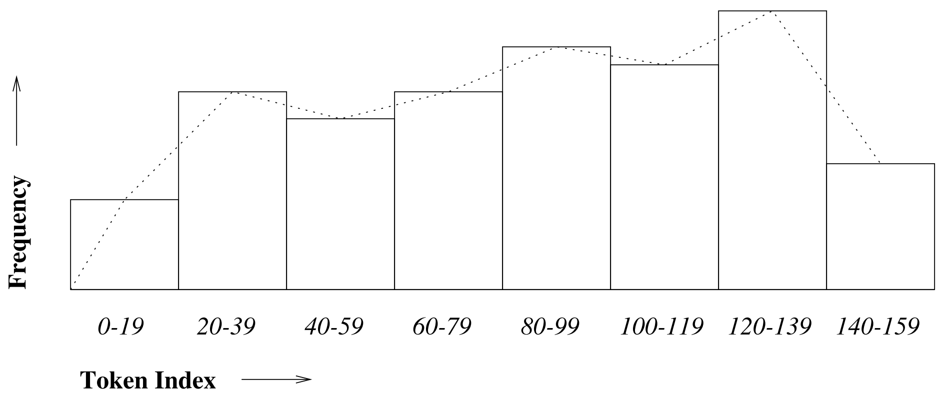

FIGURE 3.2: A depiction of the histogram used by Bayes to estimate the position likelihood of test instances.

and $\Pr(H)$? In the derivation that follows, D stands for the contents of the document. $\Pr(H_{p,k})$, therefore, is our belief in a hypothesized fragment before we have actually examined the contents of the document in which it occurs. $\Pr(D|H_{p,k})$ is the probability of seeing the contents of the document from the point of view of the fragment.

3.2.2 Derivation of Bayes

Given a document, in order to specify a hypothesis by this Bayes' Rule learner, which I call Bayes, two parameters must be set: position and length. The prior $\Pr(H_{p,k})$ comes from some distribution defined over these two parameters. The distribution used by Bayes is based on the positions and lengths of field instances as observed in the training documents. Bayes treats these two parameters as independent, modeling each separately. In Bayes, therefore, the prior belief in a hypothesis is:

$$\Pr(H_{p,k}) = \Pr(\text{position} = i)\Pr(\text{length} = k)$$

Bayes bases each of the constituent estimates on the training data.

In a typical information extraction problem field instances are short and do not vary much in length. Thus, simply tabulating the number of times instances of length k are seen during training and dividing this number by the total number of training instances yields a good estimate for the length prior. Let n be the total number of field instances seen in a training set, and let $L(k)$ represent the number of instances of length k. The length estimate used by Bayes is simply $L(k)/n$.

In order to estimate the position prior, Bayes sorts training instances into n bins, based on their position, where n is much smaller than the typical document size. During testing, Bayes uses a frequency polygon drawn over these bin counts to estimate the position prior. Figure 3.2 depicts this graphically. Each training instance is sorted into the appropriate bin by start position. The probability of seeing a test instance beginning at some position is calculated by interpolating between the midpoints of the two closest bins (the dotted line).

Bayes bases its estimate of the conditional data likelihood ($\Pr(D|H_{p,k})$) on the tokens it observes in and near the hypothesized fragment. Before Bayes is run, the user must set the parameter w, which specifies how many tokens on either side of a fragment to consider in making this estimate. Tokens farther away than w tokens from the beginning or end of a fragment are then disregarded.

Each token occurring inside and up to w tokens away from the hypothesized instance contributes to the estimate of $\Pr(D|H_{p,k})$. Bayes assumes that each such contribution is independent of all the others. For each such token t, Bayes estimates the likelihood of seeing t at its particular position with respect to the fragment. Its estimate of the conditional data likelihood is a product of such individual token estimates:

$$\Pr(D|H_{p,k}) = \prod_{p-w \leq i \leq p+k+w-1} \Pr(t_i|H_{p,k})$$

In practice, the probability $\Pr(t_i|H_{p,k})$ is estimated in one of two ways, depending on whether t_i occurs within the fragment or in its context. Let us posit a set of random variables, before_j and after_j, where $1 \leq j \leq w$. The variable before_j will model the distribution of tokens observed in the jth position before any field instance in the training set. The variables after_j have the symmetric meaning; each such variable models the distribution of tokens occurring in a position following field instances. The actual conditional data likelihood estimate returned by Bayes has the following form:

$$\Pr(D|H_{p,k}) = [\prod_{j=1}^{w} \Pr(\mathit{before}_j = t_{p-j})][\prod_{j=1}^{k} \Pr(\mathit{in} = t_{p+j-1})][\prod_{j=1}^{w} \Pr(\mathit{after}_j = t_{p+k+j-1})]$$

In contrast with *before* and *after*, each of which corresponds to a set of variables, *in* is a single variable representing the distribution of tokens occurring anywhere within a field instance. The reason for this difference is the variability of instance lengths. If the instances of a particular field tend to be three tokens long, but one or two training instances are observed that consist of four tokens, and if in-field estimates are handled in a position-dependent manner, then the statistics for the fourth position may be noisy because of the very low frequency of occurrence. In contrast, every field instance has w tokens occurring before and after it, so the each of the variables before_i and after_i can be modeled with relative reliability.[2]

The probability $\Pr(\mathit{before}_j = t_i)$ is estimated as the number of times t_i occurred as the jth word before any training field instance, divided by the total number of training field instances. Similarly, $\Pr(\mathit{in} = t_i)$ is estimated as the number of times t_i occurred as a training field instance, divided by the total number of training field instance tokens. To compensate for low-frequency events, m-estimates are used as part of all such calculations (Cestnik, 1990).

[2]Bayes inserts placeholder tokens for field instances occurring near a document boundary.

```
Procedure BayesAccount(doc, fieldname)
   fbounds = FieldInstanceBounds(doc, fieldname)
   For (firsti, lasti) in fbounds   /* for each index pair */
   PositionAccount(firsti)           /* For position prior */
   LengthAccount(lasti - firsti + 1) /* For length prior */
   /* Update in */
       For i = firsti to lasti
          token = TokenAt(doc, i)
          in{token} = in{token} + 1
       End For
       For i = 1 to $w$
    /* Update before */
          tab = before[i]
          index = firsti - i
          token = TokenAt(doc, index)
          tab{token} = tab{token} + 1
    /* Update after */
          tab = after[i]
          index = lasti + i
          token = TokenAt(doc, index)
          tab{token} = tab{token} + 1
       End For
    End For
    /* Update all */
    For i = 0 to LastTokenIndex(doc)
       token = TokenAt(doc, i)
       all{token} = all{token} + 1
    End For
 End Procedure
```

TABLE 3.2: The training procedure used by all Bayes variants.

```
1 Function BayesEstimate(doc, firsti, lasti)
2    logprob = log(PositionPrior(firsti))
3            + log(LengthPrior(lasti - firsti + 1))
4    For i = 1 to $w$
5       tab = before[i]
6       token = TokenAt(doc, firsti - i)
7       count = tab{token}
8       logprob = logprob + log(MEst(count, totalFIcount))
9    End For
10    For i = firsti to lasti
11       token = TokenAt(doc, i)
12       count = in{token}
13       logprob = logprob + log(MEst(count, totalFieldTokens))
14    End For
15    For i = 1 to $w$
16       tab = after[i]
17       token = TokenAt(doc, lasti + i)
18       count = tab{token}
19       logprob = logprob + log(MEst(count, totalFIcount))
20    End For
21    Return logprob
22 End Function
```

TABLE 3.3: Bayes's estimating procedure for text fragments.

Training is a matter of scanning the training corpus and building the various frequency tables needed for Bayes's estimates. Table 3.2 presents the training procedure for Bayes, as well as variants of the algorithm described below. The procedure works by side effect on the global variables `in`, `before`, `after`, and `all`.[3] Both `in` and `all` represent hash tables mapping tokens to frequency counts; `before` and `after` are *arrays* of such tables. The function `TokenAt(doc, i)` (e.g., line 11) returns the token occurring at position i in the document, unless i is out of bounds, in which case it returns a placeholder. Note that, for the sake of clarity, this pseudocode depicts Bayes's training procedure as making two passes through a document, once to update `in`, `before`, and `after`, and once to update `all`. In fact, it is straightforward to perform all necessary accounting in a single pass.

During testing, an estimate is produced for every fragment in a document of a size having non-zero probability (i.e., a size actually seen in training). Table 3.3 shows the estimating procedure for Bayes. The global variables `totalFIcount` (lines 8 and 19) and `totalFieldTokens` (line 13) hold the number of field instances and field instance tokens seen during training, respectively. The function `MEst(num, den)` (lines 8, 13, and 19) returns an m-estimate, where `num` represents the numerator, `den` the denominator of the desired ratio. The position and length prior estimates are returned by the functions `PositionPrior(startindex)` (line 2) and `LengthPrior(length)` (line 3), respectively.

Table 3.4 shows a sample prediction for $w = 4$. Tokens listed above the phrase in the

[3]The variable `all` is not used by Bayes, but by a variant described below. It is included here for convenience.

Token	*Log Prob.*	*Combined*	
`00`	-1.89	-4.98	Data Likelihood
`PM`	-1.26		
`Place`	-1.04		
`:`	-0.79		
Baker	-3.01	-11.11	
Hall	-1.92		
Adamson	-3.09		
Wing	-3.09		
`Host`	-2.53	-12.37	
`:`	-0.96		
`Hagen`	-4.44		
`Schempf`	-4.44		
Position		-5.31	Prior
Length		-3.02	
Posterior		-36.79	

TABLE 3.4: A sample Bayes fragment likelihood estimation for a location phrase ("`Baker Hall Adamson Wing`") taken from the seminar announcement collection.

Token column are those occurring before it in the text, while those below occur after. The estimate of -36.79 is quite high.

Bayes discards all estimates that are below a threshold T, which is determined heuristically, as follows: Initial training is followed by a second pass through the training collection during which all field instances are re-examined. Let $P(f)$ be Bayes's estimate for some field instance f, and let F be the set of all instances in the collection. Bayes's prediction threshold is set to:

$$T = \alpha \min_{f \in F} P(f)$$

where α is a parameter set by the user in advance. If no fragment in a test document leads to an estimate above this threshold, Bayes declines to issue a prediction. Because Bayes produces its estimates as log probabilities (i.e., large negative numbers), increasing α causes Bayes to issue more predictions.

3.2.3 Modifications

Unless an extraction problem is characterized by field instances whose lengths do not vary much, we may encounter a problem with Bayes. Bayes's estimate is effectively a single large product of individual estimates. Because each token contributes an estimate, the number of terms in the product grows and shrinks with the size of the fragment under consideration. What is more, more terms also means more pessimistic estimates; therefore, longer fragments are to some extent handicapped in relation to shorter ones.

```
1 Function BayesLNEstimate(doc, firsti, lasti)
2    logprob = log(PositionPrior(firsti))
3            + log(LengthPrior(lasti - firsti + 1))
4    For i = 1 to $w$
5       tab = before[i]
6       token = TokenAt(doc, firsti - i)
7       count = tab{token}
8       logprob = logprob + log(MEst(count, totalFIcount))
9    End For
10    probsum = 0
11    probcount = 0
12    For i = firsti to lasti
13       token = TokenAt(doc, i)
14       count = in{token}
15       probsum = probsum + log(MEst(count, totalFieldTokens))
16       probcount = probcount + 1
17    End For
18    logprob = logprob + avgFIlength * probsum / probcount
19    For i = 1 to $w$
20       tab = after[i]
21       token = TokenAt(doc, lasti + i)
22       count = tab{token}
23       logprob = logprob + log(MEst(count, totalFIcount))
24    End For
25    Return logprob
26 End Function
```

TABLE 3.5: BayesLN's estimating procedure for text fragments.

BayesLN is a modification of Bayes that compensates for variations in length. Instead of a product, the estimate for in-field tokens in BayesLN is the mean of the individual token estimates multiplied by the mean length of training instances. Table 3.5 shows the procedure used by BayesLN to produce fragment estimates. The modification to Bayes occurs between line 10 and 18. Taking the mean for the in-field estimate ensures that two fragments of differing lengths receive a fair comparison. If just the mean were taken without any further adjustments, however, the contextual estimates (*before* and *after*) would receive disproportionate emphasis. With $w = 4$, we would have eight terms corresponding to fragment context and just a single term to represent the tokens found inside a fragment. Multiplying the in-field estimate by the mean instance length assigns it its appropriate weight in the larger estimate.

The experiments section will show that BayesLN is an improvement over Bayes, but it still has a weakness that becomes obvious with experimentation: Both Bayes and BayesLN assign too much weight to common tokens. For example, the token "`.`" (period) is one of the most common constituents of the *speaker* field in seminar announcements (as part of abbreviations, such as "Dr." or with middle initials). Thus, Bayes and BayesLN assign it a high estimate whenever it is encountered within a candidate fragment. Of course, this token is very common in general, something which neither of our variants takes into account. Consequently, the contents of the three-token fragment "`. . .`" (ellipsis) contribute

```
1 Function BayesIDFEstimate(doc, firsti, lasti)
2    logprob = log(PositionPrior(firsti))
3            + log(LengthPrior(lasti - firsti + 1))
4    For i = 1 to $w$
5       tab = before[i]
6       token = TokenAt(doc, firsti - i)
7       count = tab{token}
8       logprob = logprob + log(MEst(count, all{token}))
9    End For
10    probsum = 0
11    probcount = 0
12    For i = firsti to lasti
13       token = TokenAt(doc, i)
14       count = in{token}
15       probsum = probsum + log(MEst(count, all{token}))
16       probcount = probcount + 1
17    End For
18    logprob = logprob + avgFIlength * probsum / probcount
19    For i = 1 to $w$
20       tab = after[i]
21       token = TokenAt(doc, lasti + i)
22       count = tab{token}
23       logprob = logprob + log(MEst(count, all{token}))
24    End For
25    Return logprob
26 End Function
```

TABLE 3.6: BayesIDF's estimating procedure for text fragments.

strongly to Bayes's belief that it is an instance of the *speaker* field.

The final variant, which I call BayesIDF, compensates for this by discrediting common tokens. It changes how estimates assigned to individual tokens are calculated. Instead of the number of training field instances or field instance tokens, the denominator used for each such calculation is the total number of times a token occurred in the training corpus. Table 3.6 shows the modified procedure. Note how the second number (the denominator) in every m-estimate differs from that used in the other two variants (e.g., in line 8). Table 3.7 shows a sample estimate on the same fragment used for Table 3.4. The change is particularly apparent in the estimates assigned to very common tokens, such as colons, and to the tokens occurring within the hypothesized instance, which are reasonably common inside instances of the *location* field but rare overall.

3.3 Experiments

In this section I present experimental results comparing Rote and the three variants of Bayes on the seminar announcement and acquisition domains. Although the two experiments differ in the details, they share the same framework. In each set of experiments, the entire document collection is randomly partitioned several times (five times with the sem-

Token	*Log Prob.*	*Combined*	
`00`	-2.63	-9.73	Data Likelihood
`PM`	-2.11		
`Place`	-1.14		
`:`	-3.85		
Baker	-0.99	-3.80	
Hall	-0.90		
Adamson	-0.99		
Wing	-1.00		
`Host`	-1.87	-10.08	
`:`	-4.02		
`Hagen`	-2.05		
`Schempf`	-2.14		
Position		-5.31	Prior
Length		-3.02	
Posterior		-31.94	

TABLE 3.7: A sample BayesIDF fragment likelihood estimation for a location phrase ("`Baker Hall Adamson Wing`") taken from the seminar announcement collection.

inar announcements, ten with the acquisitions articles) into two sets of equal size, training and testing. The learners are trained on the training documents and tested on the corresponding test documents for each such partition. The resulting numbers are averages over documents from all test partitions.

3.3.1 Case Study: Seminar Announcements

The seminar announcement experiments are designed to answer three questions. First, I am interested in the comparative performance of the three variants of Bayes. The comparison will show that BayesIDF performs best on all four fields. Second, I want to determine how well Rote measures up to BayesIDF. And finally, of course, the experiments should provide some insight into the suitability of these approaches as standalone extractors for the kind of text genre that is a central focus of this dissertation—informally constructed text.

Table 3.8 shows the precision achieved by each learner at maximum recall, the *Prec* column listing precision and the *Rec* column listing recall. It is important to keep in mind the interaction between these two numbers. While Rote, for example, achieves a surprising 55.1% precision on the speaker field, which compares very favorably with BayesIDF's performance, this score is at much lower recall.

Table 3.9 compares precisions at approximately 25% recall. Missing values indicate a learner did not achieve 25% recall. Note that, depending on the distribution of confidence

	speaker		*location*		*stime*		*etime*	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
Bayes	10.0	11.8	32.8	34.3	96.2	96.2	42.6	91.7
BayesLN	11.5	13.6	44.8	46.9	98.1	98.1	44.4	95.6
BayesIDF	28.8	27.4	57.3	58.8	98.2	98.2	46.8	95.7
Rote	55.1	6.8	89.5	58.1	73.7	73.4	37.4	95.7

TABLE 3.8: Precision and recall of Rote and three variants of Bayes on the four seminar announcement fields.

	speaker		*location*		*stime*		*etime*	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
Bayes	—	—	50.7 ± 4.1	25.0	100.0 ± 0.0	25.1	100.0 ± 0.0	25.7
BayesLN	—	—	93.9 ± 2.7	25.0	100.0 ± 0.0	25.1	100.0 ± 0.0	25.0
BayesIDF	35.6 ± 3.5	25.0	97.7 ± 1.7	25.2	100.0 ± 0.0	25.3	100.0 ± 0.0	25.0
Rote	—	—	99.2 ± 1.2	24.8	78.2 ± 4.0	26.3	79.4 ± 5.7	27.3

TABLE 3.9: Precision at the approximate 25% recall level of Rote and the three variants of Bayes on the four seminar announcement fields.

scores, it is not always possible to choose a confidence cut-off, such that exactly 25% recall is attained. This explains the occasional slight variations in recall.

The peak F1 scores shown in Table 3.10 reduce the comparison of learners to a single number. The *F1* column lists the maximum F1 score achieved by a learner at any point on its precision/recall curve, and *Prec* and *Rec* show the corresponding precision and recall, respectively. Recall that an F1 score in bold face shows the learner judged best with 95% confidence. Here we see why it is important to examine performance at less than full

	speaker			location			stime			etime		
	F1	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*
Bayes	12.0	14.8	10.1	36.3	41.9	32.0	96.2	96.2	96.2	85.5	97.5	76.1
BayesLN	14.8	22.3	11.0	48.2	53.2	44.0	98.1	98.1	98.1	88.7	83.9	94.1
BayesIDF	**29.7**	41.8	23.0	61.3	66.3	57.0	98.2	98.2	98.2	**92.3**	94.6	90.1
Rote	12.1	55.1	6.8	**70.6**	90.1	58.1	73.9	74.8	73.0	53.6	53.1	54.1

TABLE 3.10: Peak F1 scores and corresponding precision and recall for Rote and BayesIDF on the seminar announcement fields.

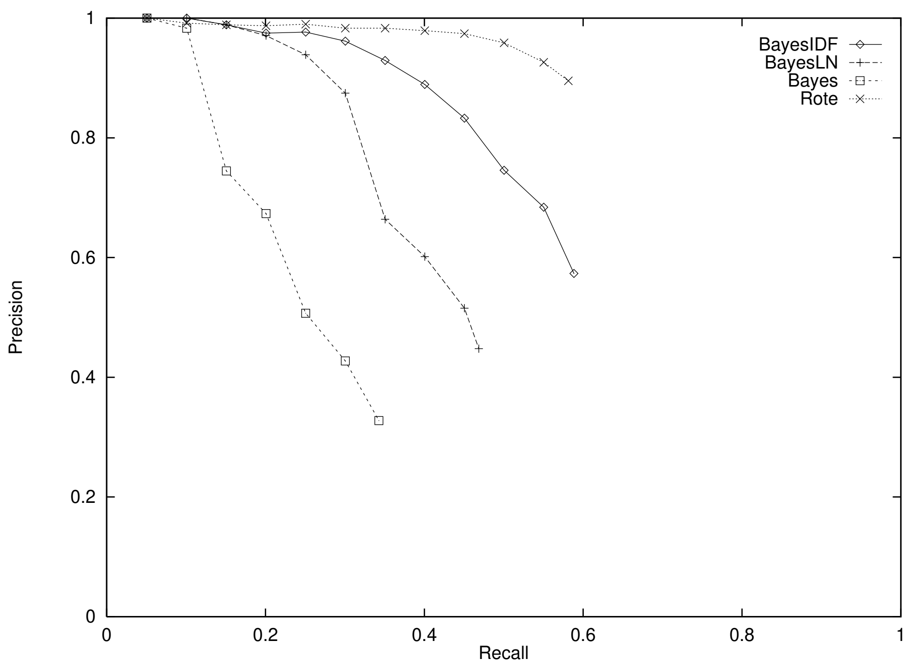

FIGURE 3.3: Precision of Rote and BayesIDF as a function of recall on the seminar *location* field.

recall. Although BayesIDF's full-recall performance on *etime*, as reported in Table 3.8, appears to leave much to be desired, the corresponding numbers in Table 3.10 show that BayesIDF actually performs quite well on this field. The relatively low frequency with which *etime* occurs—approximately in only half of the documents—accounts for the discrepancy. BayesIDF's low precision at full recall in Table 3.8 is due to a large number of spurious predictions, but Table 3.10 shows that BayesIDF is able, by means of low confidence scores, to separate these spurious predictions from correct ones.

Without exception, BayesIDF achieves precision, recall, and F1 scores that are at least as good as either of the other Bayes variants. The fact that BayesLN also consistently scores higher than Bayes suggests that the performance improvement is attributable to both length normalization and modified term estimates. Together, these two modifications make for a learner, BayesIDF, that is to be preferred over the other two variants.

Rote is such a simple approach that its performance can give us insights into certain characteristics of a domain. From Rote's performance on the speaker field we can infer that it is not common for the names of speakers to appear in multiple documents. And in only about half of the cases where this occurs does the re-appearance correspond to the return engagement of a speaker.

Rote's performance on the location field is truly surprising. From its 25%-recall performance presented in Table 3.9 we can safely conclude that it is more precise in identifying seminar locations than BayesIDF—for the subset of documents to which it is applicable.

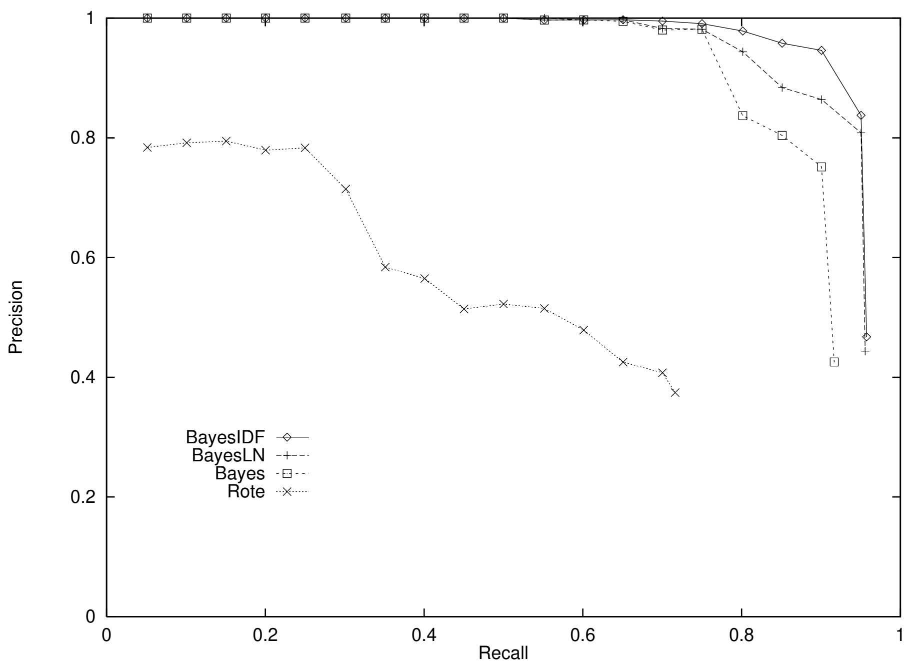

FIGURE 3.4: Precision of Rote and BayesIDF as a function of recall on the seminar *etime* field.

Figure 3.3, which presents the precision/curves for all learners on the *location* field, bolsters this impression. A little reflection makes clear why Rote performs so well on this task. University departments tend to designate certain locations for seminars and lectures, and the name of such a location (e.g., "Wean Hall 5409") tends not to occur in any other context than as the location of such a meeting.

Rote's performance on the two "time" fields illustrates its limitations. In contrast with locations, times occur frequently in this collection. Certain times are common as start and end times, a phenomenon that allows Rote to disambiguate some of these occurrences, but in order to identify instances of these fields reliably, attention to context is critical. As it happens, instances of these fields tend to occur in stereotypical contexts, a fact that all variants of Bayes are good at exploiting.

Figure 3.4 makes this clear and shows why it is useful to use precision/recall graphs in assessing a learner. It is evident from this figure that the poor full-recall precision of all Bayes variants misrepresents their ability to extract seminar end times. In particular, BayesIDF performs at a high level of precision for about 90% of the documents containing instances of *etime*.

In contrast with *etime*, BayesIDF's performance on *stime* requires no tweaking of its prediction threshold: Its mastery of this field is nearly complete. This bespeaks a high regularity in language surrounding instances of this field and a high frequency of occurrence in the data set (*stime* is instantiated at least once in every document). It is typical, for ex-

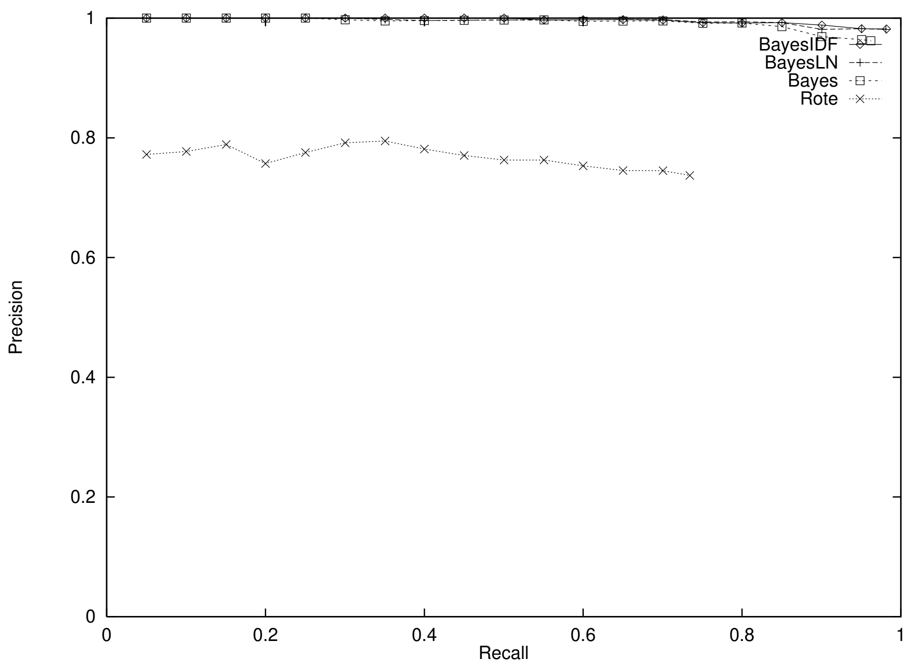

FIGURE 3.5: Precision of Rote and BayesIDF as a function of recall on the *stime* field.

ample, for a start time to be prefixed with the label `Time:`, and all variants of Bayes excel at identifying such superficial patterns.

Although a small fraction of *speaker* instances have sufficient regularity to allow these two learners to identify them, they are generally much harder to find than those of the other three fields. Figure 3.6 shows how precision drops off sharply with increasing recall. All learners are bedeviled by the relative rarity of most speaker tokens. The difficulties of Bayes and its variants are compounded by variability in the context of speaker instances. While some speaker instances are preceded by regular labels, many more occur in grammatical contexts, or in contexts employing layout clues. Some of these patterns can be observed in Appendix B, where sample seminar announcements are presented.

3.3.2 Case Study: Acquisitions

Documents in the acquisitions collection are quite different from the seminar announcements. Rather than informal, telegraphic language with a preponderance of suggestive labels, the documents in this collection contain prose written to a journalistic standard. There are a total of ten fields in this domain:

- **acquired** the official name of the company or resource in the process of being acquired
- **purchaser** the official name of the purchaser

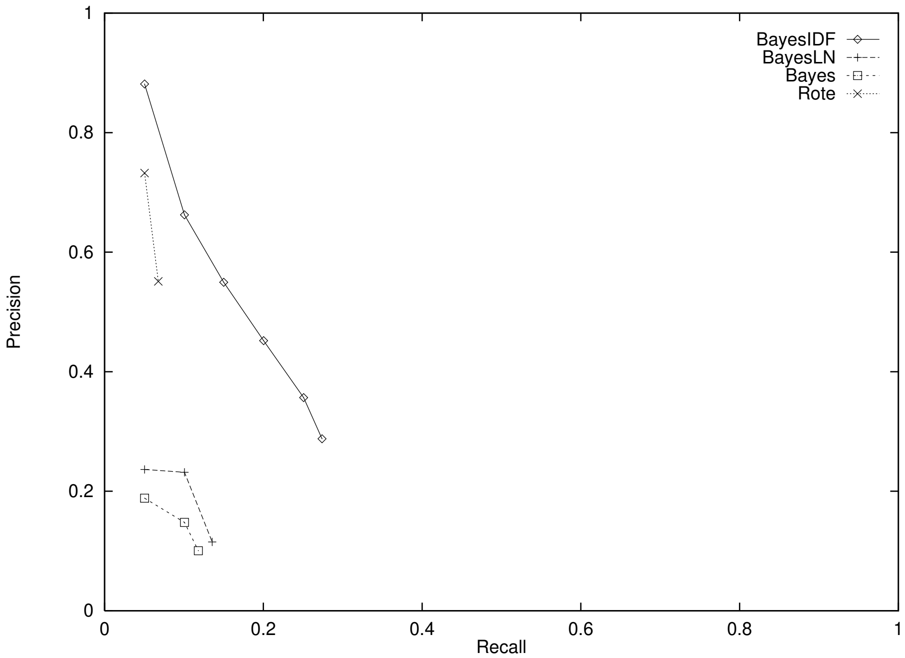

FIGURE 3.6: Precision of Rote and BayesIDF as a function of recall on the seminar *speaker* field.

- **seller** the official name of the seller
- **acqabr** the short form of *acquired*, as typically used in the body of the article (e.g., "IBM" for "International Business Machines Inc")
- **purchabr** the short form of the purchaser
- **sellerabr** the short form of the seller
- **dlramt** the amount paid for the acquisition
- **status** a short phrase indicating the status of negotiations
- **acqloc** the geographical location of *acquired*
- **acqbus** the business of *acquired* (e.g., "bank" or "software for home entertainment")

Performance numbers presented below are the result of a 10-fold experiment in this domain. The object of this experiment, which compares Rote and BayesIDF, is to determine to what extent the encouraging results observed for the seminar announcements carry over to a more traditional information extraction problem.

If the term-space learners are able to make inroads into most of the seminar announcement fields, the situation is reversed in the acquisitions domain. Table 3.11 shows precision

	Approx. 25% recall				*Full recall*			
	Rote		BayesIDF		Rote		BayesIDF	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
acquired	—	—	—	—	59.6	11.2	19.8	20.0
purchaser	—	—	52.2 ± 2.7	25.0	43.9	10.8	36.9	40.4
seller	—	—	30.6 ± 2.9	25.0	41.7	10.8	15.6	38.7
acqabr	—	—	36.4 ± 2.4	25.0	22.1	12.0	23.2	32.1
purchabr	—	—	51.7 ± 3.0	25.1	16.8	9.4	39.6	52.9
sellerabr	—	—	33.0 ± 3.5	25.0	9.8	7.8	16.0	51.5
dlramt	74.3 ± 3.9	26.7	75.5 ± 4.0	25.0	63.2	38.8	24.1	54.5
status	67.8 ± 3.2	24.6	51.5 ± 2.9	25.0	42.0	50.7	33.0	43.6
acqloc	—	—	—	—	6.4	12.4	7.0	23.6
acqbus	—	—	—	—	8.2	6.7	4.1	10.7

TABLE 3.11: Precision of Rote and BayesIDF on the ten acquisitions fields at two recall levels, 25% and full.

	acquired			purchaser			seller					
	F1	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*			
Rote	18.9	66.5	11.0	17.4	43.9	10.8	17.2	41.7	10.8			
BayesIDF	20.2	21.7	19.0	**39.5**	40.0	39.0	**28.5**	28.9	28.0			
	acqabr			purchabr			sellerabr					
Rote	17.0	37.5	11.0	13.5	26.4	9.0	9.8	15.7	7.1			
BayesIDF	**29.8**	34.8	26.0	**47.4**	47.8	47.0	**31.8**	29.1	35.1			
	dlramt			status			acqloc			acqbus		
Rote	48.7	67.4	38.1	**49.6**	50.3	49.0	10.3	10.6	10.0	7.4	8.2	6.7
BayesIDF	**52.6**	58.2	48.0	41.3	43.9	39.0	**20.7**	24.1	18.1	9.0	9.0	9.0

TABLE 3.12: Peak F1 scores, with corresponding precision and recall, for Rote and BayesIDF on the acquisitions fields.

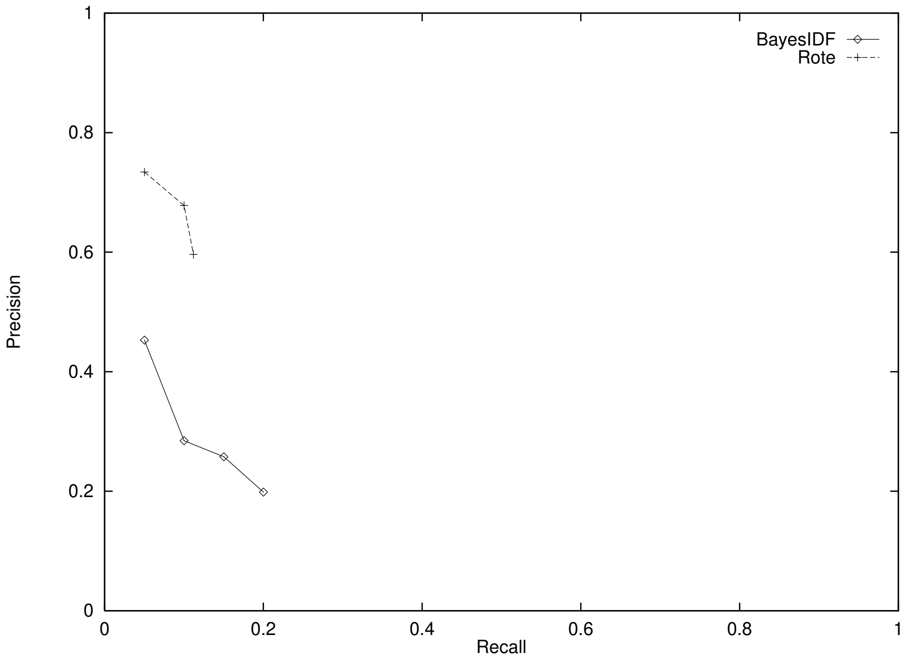

FIGURE 3.7: Precision of Rote and BayesIDF as a function of recall on the *acquired* (purchased company or resource) field.

and recall performance of Rote and BayesIDF in this domain, both at approximately 25% recall and at the maximum recall achieved by the learner. Again, missing values indicate that a learner did not achieve 25% recall for the respective field. Table 3.12 presents the corresponding F1 scores. The most striking feature of these numbers in both these tables is how much harder the acquisition fields are than the seminar announcement fields. The same comparative pattern is also evident: Rote has limited applicability, achieving competitive performance on a couple fields—*status* and *dlramt*—while BayesIDF achieves higher recall scores.

Figures 3.7 through 3.11 present precision/recall comparisons of Rote and BayesIDF on five of the acquisitions fields. Figure 3.8 (*purchaser*) can be regarded as typical: BayesIDF achieves higher precision than Rote at comparable recall levels and higher recall overall. Precision/recall curves due to BayesIDF also tend to exhibit a more or less smooth, monotonic decline, indicating a learner whose confidence correlates well with the probability that a prediction is correct. Typically, of course, the decline is too steep for BayesIDF to be very useful by itself. Rote, on the other hand, often does not achieve high enough recall levels to make a comparison of curves between the two learners interesting.

The difference in performance for BayesIDF between the *acquired* (Figure 3.7) and *purchaser* (Figure 3.8) fields is somewhat surprising. The two fields have roughly the same frequency of occurrence—*acquired* and *purchaser* occur on average 1.14 and 1.04 times per file, respectively—and both have the same typical surface form (i.e., names of compa-

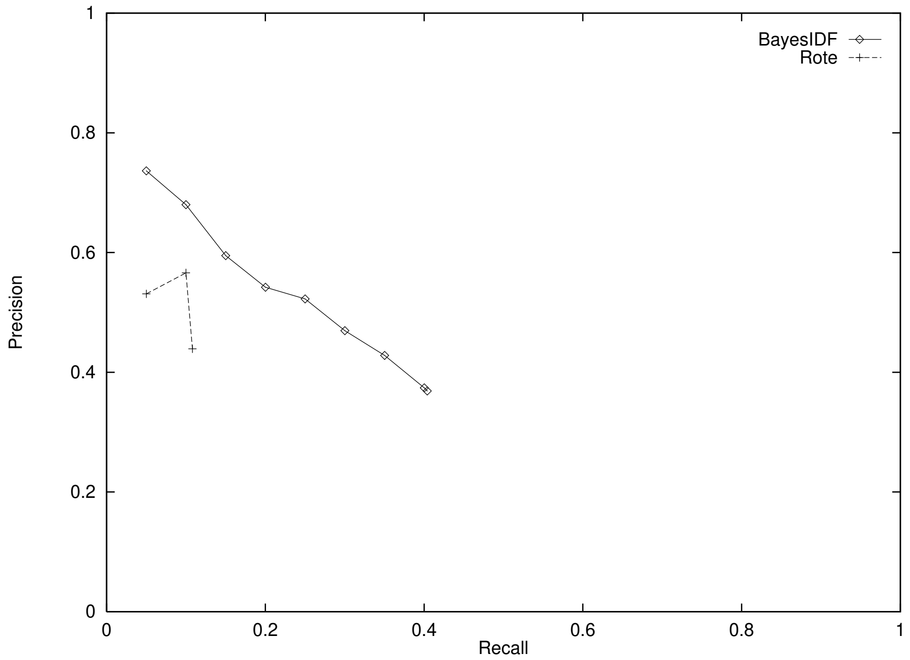

FIGURE 3.8: Precision of Rote and BayesIDF as a function of recall on the *purchaser* field.

```
FIRST WISCONSIN <FWB> TO BUY MINNESOTA BANK
 MILWAUKEE, Wis., March 26 - First Wisconsin Corp said it
plans to acquire Shelard Bancshares Inc for about 25 mln dlrs
...
```

TABLE 3.13: The first three lines of an acquisition article showing a typical pattern of field instantiation: *purchaser* immediately following the dateline.

nies). I attribute this difference to conventions of presentation which BayesIDF is better able to exploit for *purchaser* than for *acquired*. It is common for the name of the purchasing company to head the lead sentence in an acquisition article, immediately following the dateline. An example of this is shown in Table 3.13. BayesIDF is able to use regularities found in the dateline and the sentence's main verb (e.g., "said" or "announced"). Similar regularities surround many instances of the acquired field, but these are apparently less common.

Figure 3.9 is typical of several of the precision/recall curves for this domain. Again, the BayesIDF curve exhibits a graceful downward trend, while Rote achieves substantially worse precision—how much worse depends on the field—and lower recall.

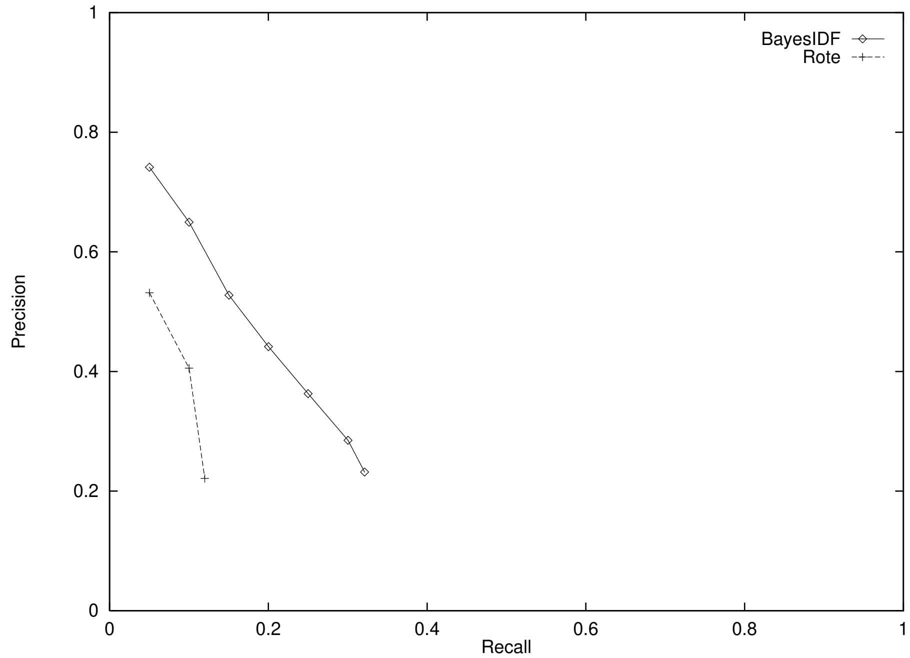

FIGURE 3.9: Precision of Rote and BayesIDF as a function of recall on the *acqabr* field (short version of *acquired*).

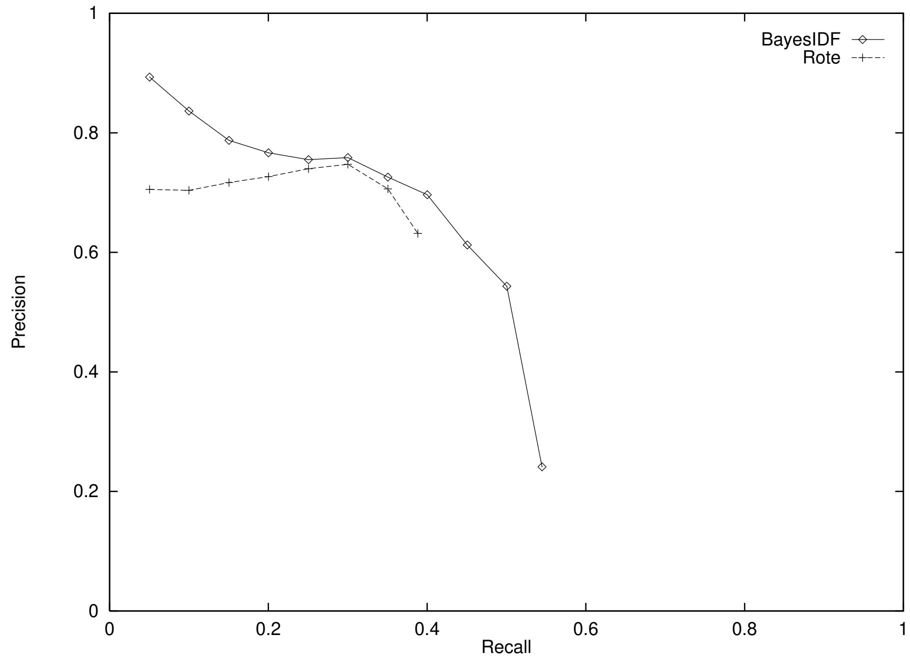

FIGURE 3.10: Precision of Rote and BayesIDF as a function of recall on the *dlramt* field.

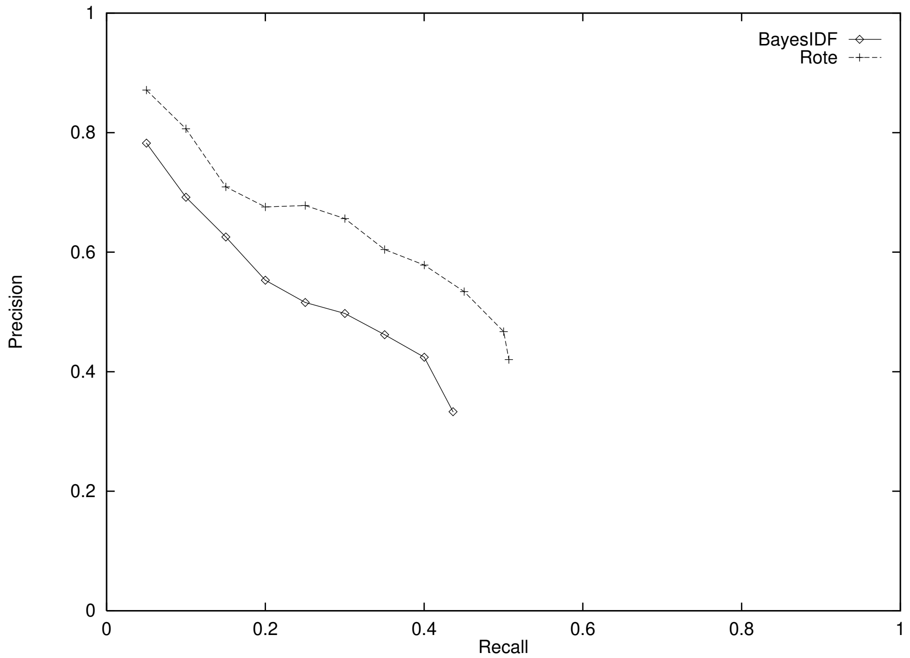

FIGURE 3.11: Precision of Rote and BayesIDF as a function of recall on the *status* field.

As with the seminar announcements, there is at least one field for which Rote achieves high enough performance to be considered useful as a standalone extractor. For the *dlramt* field (Figure 3.10) it is competitive with BayesIDF, up to a certain recall level, and for the *status* field (Figure 3.11) it is strictly better. Again, these are fields, instances of which are often easy to distinguish from the rest of the text. Most of Rote's precision on the *dlramt* field is probably attributable to its recognition of the terms "undisclosed" and "not disclosed," which instantiate this field when a company declines to reveal the price of a purchase, rather than of actual amounts paid. The words "disclosed" or "undisclosed" occur a total of 115 times in the document collection. 87 of these occurrences are as part of a *dlramt* field, accounting for about 31% of all 282 instantiations of *dlramt*. Such simple, stereotypic language is even more common for the *status* field, where phrases like "letter of intent" and "agreed to buy" are the norm.

3.4 Discussion

The results presented in this chapter leave little doubt that term-space learners like Rote and BayesIDF are appropriate for some information extraction tasks. They also make clear that their application is limited to fields characterized by highly stereotypic language. Such fields do occur naturally, as part of reasonable domain definitions; three of the four fields in the seminar announcement domain are susceptible to term-space methods. And even in "harder" domains, some fields can be handled, at least in part, by these methods (witness the *dlramt* and *status* fields in the acquisitions domain).

Not surprisingly, of the two term-space approaches, the statistical approach, Bayes, typically attains higher recall. For best performance, however, I found a straightforward adaptation of Naive Bayes needed to be modified with heuristics which, while they make sense intuitively, are hard to justify in strict Bayesian terms. Nevertheless, BayesIDF provides some indication of the kind of performance we might expect from statistical term-space learners, and I use it in the rest of the thesis both as a convenient baseline, as well as a point of departure for the experiments with grammatical inference presented in Chapter 4.

BayesIDF is strongest on fields with highly regular contexts, such as simple labels or patterns of language marked by domain-specific conventions. On the other hand, it is hampered by low-frequency tokens. For example, its performance is relatively poor on name fields (company names, the names of seminar speakers): While fields like *stime* are characterized by reasonably high-frequency tokens (e.g., "`3`", "`:`", "`00`"), the constituent tokens of name fields are as often as not relatively rare (e.g., "`Freitag`").

Rote is even more extreme in its reliance on high frequencies. It requires that whole fragments be repeated. However, some fields do have this character. For such fields Rote's precision is typically high, and its confidence scores are relatively reliable, even if it achieves lower recall than alternative approaches. Because of this, it is attractive as a partial solution to a field extraction problem in many cases.

Rote's reasonable performance on a few of the fields these experiments investigate prompts an important question: Just how hard are these learning tasks? The nearly perfect performance of BayesIDF on the seminar start time field may give rise to similar concerns. If a knowledge-poor learning approach can solve a task, perhaps it would be more fruitful to look elsewhere for interesting learning problems. Certainly no slot-filling problem studied at MUC can be solved so easily.

In fact, it is difficult to know this for certain by analyzing results reported in the MUC proceedings. The MUC performance numbers represent *average* performance over all slots either predicted by a system or present in the answer keys. Thus, they shed no light on the difficulty of filling a particular type of slot. This is a reflection, in part, of goals that differ from the ones I have set for myself in this thesis. As noted in the previous chapter, a MUC system is a collection of components devoted to the completion of diverse tasks. MUC evaluations measure a system's ability to screen out irrelevant documents, spot candidate noun phrases, combine evidence from different locations in a document, in addition to filling individual slots. Consequently, a comparison between the results I report and those reported for MUC tasks is somewhat dubious. And, as stated, my aim is less the design of an end-to-end information extraction system for journalistic or technical prose than an investigation into learning approaches to the slot filling problem.

Aside from the problem of extracting from HTML, little attention has been paid to "unconventional" information extraction problems—far less attention that the potential usefulness of good solutions would seem to warrant. Thus, I account BayesIDF's performance on the *stime* field, an admittedly easy learning task, a success. Finding a seminar start time is in some sense a “natural” problem, and a type of problem which has been explored very little.

The argument that Rote is a priori too simple can be met with similar considerations. In fact, as we have seen, Rote is useful for some natural extraction problems. And even if its usefulness is limited to a fraction of the documents in a domain, a real information extraction system can realize benefit by treating it as a source of information to be considered in making final extraction decisions.

Chapter 4

Learning Field Structure with Grammatical Inference

> The limitations of BayesIDF are apparent in some of the errors in makes, errors of boundary identification. BayesIDF forms its estimate based on term frequency statistics and lacks any notion of abstract *structure*. In this chapter I ask if it is possible to graft a notion of structure onto BayesIDF by combining it with a learner that *can* recognize structure. This is an application of *multistrategy learning*. I review the paradigm of *grammatical inference* and describe Alergia, a prominent algorithm from this paradigm. I also discuss the problem of representing text fragments so that effective generalization can occur. The proposed solution to this representation problem is a covering algorithm for inferring decision lists, which are used to transduce raw fragments into an abstract form. Experiments in which BayesIDF is combined with grammars learned over transduced field instances show large improvements in performance over that achieved by either BayesIDF or the grammars in isolation.

Although BayesIDF is surprisingly good at identifying field instances, it can have difficulty identifying boundaries precisely. Often, BayesIDF extracts an unintelligible piece of an instance, or a fragment containing tokens from the surrounding text which a human would consider trivial to filter out. Figure 4.1 gives examples of some BayesIDF predictions which, though counted as errors, are successful at approximately locating field instances. Each prediction in the figure is BayesIDF's highest-confidence prediction for some document. How well would BayesIDF have performed if such responses were not counted as errors? Figures 4.2 and 4.3 attempt to answer this question, showing precision/recall curves for BayesIDF under three separate criteria—*overlap*, *contain*, and *exact*. *Overlap* counts a prediction correct if any part of it shares a token with a field instance. *Contain* counts a prediction correct if it contains all tokens of a field instance, plus at most 5 neighboring tokens. The comparison is striking. It is evident that for a substantial number of documents BayesIDF finds a field instance without getting its boundaries right. In-

Location

```
Confidence:    -89.69
Fragment:      GSIA 259 Refreshments served

Confidence:    -77.30
Fragment:      Mellon Institute.
```

Speaker

```
Confidence:    -68.97
Fragment:      Dr.

Confidence:    -80.84
Fragment:      Antal Bejczy Lecture Nov. 11
```

FIGURE 4.1: Examples of poor alignment from actual tests of BayesIDF.

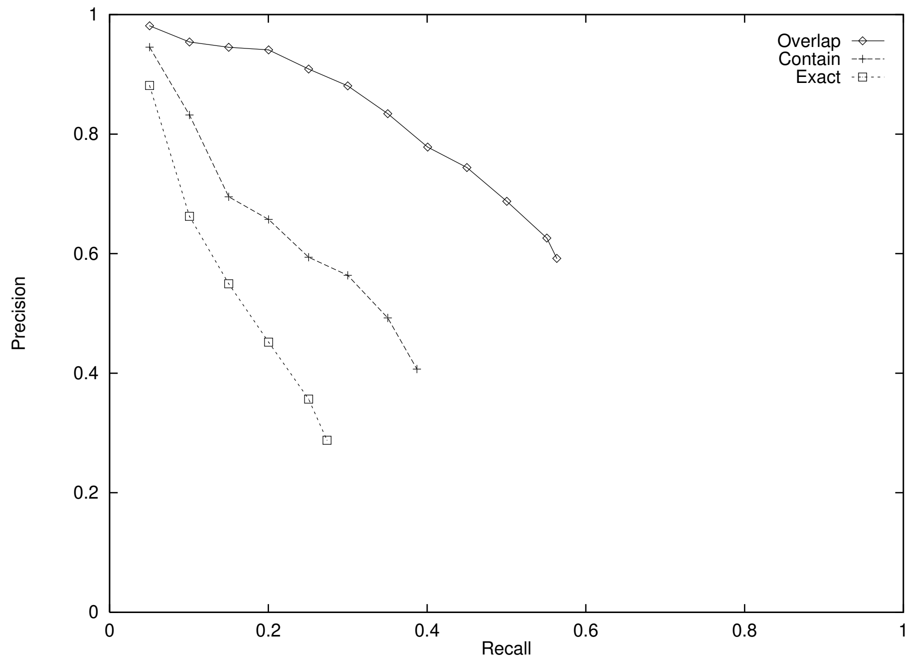

FIGURE 4.2: Effect of changing criterion of correctness on BayesIDF performance on the *speaker* field.

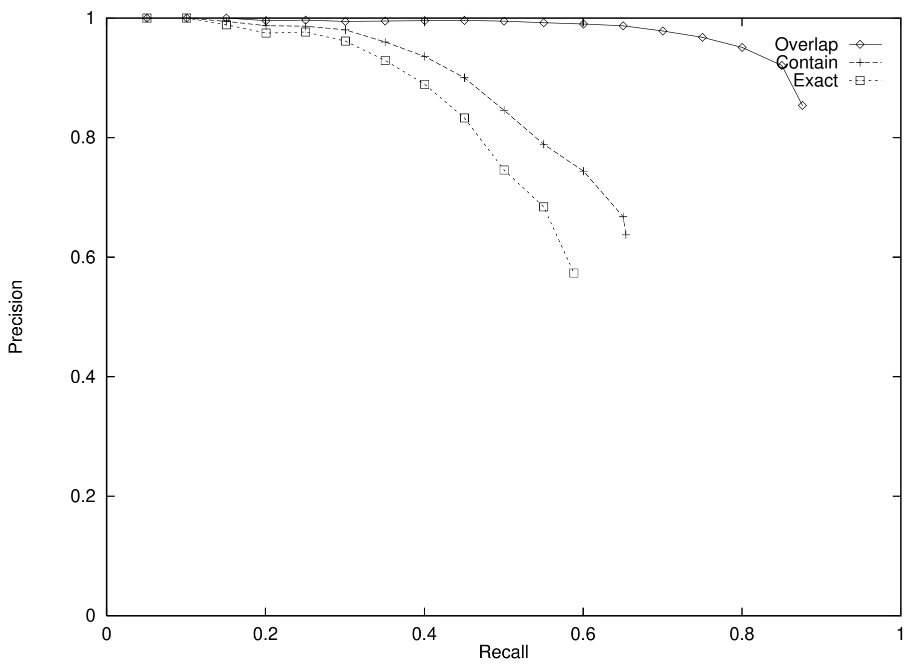

FIGURE 4.3: Effect of changing criterion of correctness on BayesIDF performance on the *location* field.

deed, it is almost perfect at identifying the approximate region where the *location* field is instantiated.

BayesIDF and Rote share a fundamental limitation: They cannot exploit abstract features of the text, but must rely on statistics based on the occurrence of raw terms. For example, they cannot express that a token belongs to the class of capitalized or numeric tokens. This gives rise to BayesIDF's alignment difficulties whenever a field instance contains or is surrounded by uncommon terms. The *speaker* fragments in Figure 4.1 make this clear. The "`Dr`" token in the top fragment is a common component of seminar speaker names; a large number of the last names encountered, however, are not so common. Consequently, BayesIDF reaches a higher estimate by excluding the last name. The result is a prediction which, taken as a whole, is useless, but not absurd. It is apparent to a human observer that BayesIDF's prediction is nearly correct, and that the important text follows the extracted fragment. The phrase "`Dr.`" sets up strong expectations to this effect. A human reader can quickly locate and extract names having this form—Dr. *capitalized-word*—even without reading a text for comprehension.

4.1 Grammatical Inference

Clearly, BayesIDF is hobbled by its lack of any notion of field structure. This section describes a remedy for this lack using *grammatical inference*. Grammatical inference refers to a class of algorithms that infer formal language grammars from example sequences. This section describes one algorithm from the literature, Alergia (Carrasco and Oncina, 1994), and proposes a way to graft it onto BayesIDF as a means of supplying BayesIDF with a notion of structure. Before this can happen, however, the text that is used to train Alergia must be represented in a suitable form, as sequences of symbols from some alphabet of manageable size. These symbols should reflect the elements of structure we want the grammar to capture. Generating this alphabet and a method for translating text into alphabet symbols—what I term *alphabet transduction*—is treated as a separate learning problem. Beginning with 26 token features, I describe and experiment with three related methods for automatically generating a transducer.

It is hard to avoid the impression that a simple notion of the appropriate structure of a field might improve the alignment of the predictions shown in Figure 4.1. Suppose we had a method which could assess a candidate field instance and return an estimate of the probability that it has the right structure, where *structure* captures some of the intuitions developed above. How might we enhance BayesIDF to take advantage of this estimate? The idea I pursue in this chapter is to add this structure estimate as another term in the product that constitutes BayesIDF's larger estimate. Recall that we are using Bayes' Rule to construct our estimate:

$$\Pr(H|D) = \frac{\Pr(D|H)\Pr(H)}{\Pr(D)}$$

where H is some hypothesis and D is data that serves to confirm or deny it. But because we only seek to maximize this formula over a set of competing hypotheses, we ignore the denominator and concentrate on maximizing $\Pr(D|H)\Pr(H)$. The solution I propose retains BayesIDF's estimate of $\Pr(H)$ (a product of position and length estimates) and seeks to refine $\Pr(D|H)$. BayesIDF's estimate is a product of numbers reflecting the occurrence of terms in and around the hypothesized field instance. I will abbreviate this product with the term $\Pr(\textit{terms}|H)$. In other words, for BayesIDF:

$$\Pr(D|H) = \Pr(\textit{terms}|H)$$

Now, suppose we have an estimate of the structural appropriateness of a hypothesized instance. In the spirit of Naive Bayes, we can augment our conditional estimate thus:

$$\Pr(D|H) = \Pr(\textit{terms}|H)\Pr(\textit{structure}|H)$$

The virtue of this approach is that it amounts to adding a single term to the larger product that already constitutes BayesIDF's estimate. BayesIDF can be run unaltered. The structural estimate, which is given to us by some independently run algorithm, is multiplied with BayesIDF's estimate to produce the estimate of what I will call BayesGI.

To make this structural estimate, I borrow ideas and an algorithm from the field of *grammatical inference*. In this section I present an outline of the grammatical inference problem and sketch the state-merging method for solving it. I also describe Alergia, a leading state-merging algorithm, which is used in the experiments presented below.

4.1.1 General Setting

In broad terms, the grammatical inference problem is this: Given a set of sequences from some formal language, induce a grammar for the language. We are given an alphabet Σ and a set of sequences S composed of symbols from Σ. The sequences in S come from some unknown language $L \subseteq \Sigma^*$. (In some settings we are also given a set of sequences S' not in L.) The object of grammatical inference is to identify L, i.e., to construct a grammar that will accept any sequence from L and reject any sequence not in L.

The tractability of this problem depends on a number of factors: the size and comprehensiveness of S, the availability of S', the class of languages from which L is drawn, and the strictness of the identification requirements, among other things. The experiments presented in this chapter assume that L is the class of regular languages, and that grammatical inference methods for learning *finite state automata* (FSA) are appropriate. There are a number of general-purpose approaches to this problem available in the literature. I consider algorithms that assume the presence of only positive training data. There are a couple of reasons for this. For one thing, although a requirement of negative data is not difficult to fulfill in this domain—there is plenty of non-field text for any given extraction problem—it introduces a sampling problem. There is such a large number of potential negative instances that it might be necessary to use only a subset, and the quality of results may depend on how the negative data are selected. For another thing, this domain does not provide any guarantee that the set of negative and positive examples are disjoint. For example, the fragment "`3:30`" appearing in a seminar announcement may or may not be the start time of a seminar (an instance of *stime*). Grammatical inference algorithms designed from formal considerations assume that a *consistent* solution is possible. Such algorithms are therefore unsuitable.

4.1.2 State-Merging Methods

Given that we believe L is a regular language, one common approach to grammar construction, which has formed the basis of a number of grammatical inference algorithms, is state merging. Starting with a maximally specific grammar (called the *canonical acceptor*), one which accepts exactly the sequences from the positive training set S and rejects all others, we proceed iteratively to merge pairs of states, thereby creating more general grammars. Thus, G_i, the grammar at time step i, has one fewer state than G_{i-1}, and G_i accepts any sequence that G_{i-1} accepts, i.e., it G_i is a generalization of G_{i-1}. Furthermore, depending on the connectivity of the merged states, G_i may accept sequences that G_{i-1} rejects. The space implicitly searched in this way constitutes a lattice, a "version space" of possible grammars

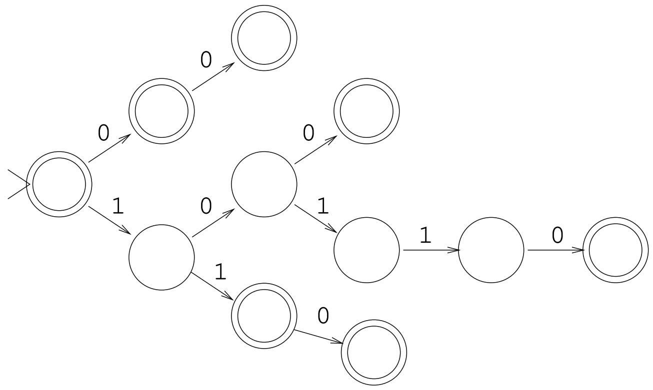

FIGURE 4.4: A canonical acceptor (prefix-tree grammar) representing the training sample $\{110, \lambda, \lambda, \lambda, 0, \lambda, 00, 00, \lambda, \lambda, \lambda, 10110, \lambda, \lambda, 100\}$.

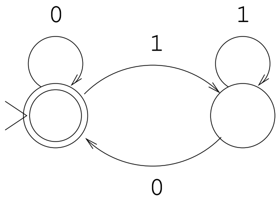

FIGURE 4.5: The grammar after merging the states of the grammar shown in Figure 4.4 using Alergia at a particular setting of its generalization parameter α.

(Dupont *et al.*, 1994; Mitchell, 1982). Note that when this search is conducted without the benefit of negative data, there can be no "G-set," no natural limit to generalization.

The canonical acceptor takes the form of a prefix tree. The prefix tree grammar is the unique deterministic tree encoding of a set of sequences, and it has the requisite feature of accepting only those sequences. Figure 4.4 shows an example borrowed from (Carrasco and Oncina, 1994), the prefix tree for a set of strings from a language built from an alphabet of 0's and 1's. Not shown are the frequencies Alergia associates with states and transitions. The double circles represent states which "accept"; only sequences produced by beginning at the start state and terminating in some accepting state belong to the language this automaton encodes. Obviously, since it accepts only the sequences in a finite sample, the canonical acceptor cannot recognize a language with infinite cardinality. However, state merging can introduce loops into the grammar (Figure 4.5), so that after generalization *any* regular language can be represented in principle.

How to choose the states to merge and when to stop merging are details left to the individual algorithm. State merging can be based on evidence local to the two states to be merged, or on some global criterion (e.g., whether the proposed merge introduces a loop, or whether it skews the distribution of state fan-out unacceptably). Stopping may depend on the availability of good merges, but ideally it will also consider some prior notion of the structure of the target language L. The difficulty of these choices is exacerbated by a reliance on exclusively positive examples. Thus, perhaps even more than related machine learning settings, grammatical inference would benefit from convenient methods for expressing prior expectations about the target language L, and for integrating those expectations into the search for a grammar. Unfortunately, the bulk of work in this area has been in the development of generic algorithms that address small, formally constrained parts of the larger problem.

4.1.3 Alergia

Alergia is a leading method for inferring stochastic finite state automata in response to positive training sequences (Carrasco and Oncina, 1994). A stochastic FSA is a generalization of a deterministic FSA in which each transition and each accepting state has an associated probability. For any given state in such an automaton, the probability of acceptance (i.e., of the state being terminal) and the probabilities of its outgoing transitions must all sum to one. Thus, a probability can be associated with any sequence belonging to the language the FSA models. This *membership probability* is the product of the transition probabilities along the unique state trajectory encoded by the sequence, and the acceptance probability of the terminal state.

In Alergia, search is organized as a single $O(n^2)$ pass through the set of states, in which, for each pair of states S_i and S_j, the question is posed, "Are S_i and S_j equivalent?" State equivalence has two components:

- The two states *accept* with the same probability.
- The two states have equivalent out-transition behavior. For any symbol in Σ, the corresponding outgoing transitions of the two states have the same probability, *and* the two states reached by following the respective transitions are equivalent.

If two states are deemed equivalent according to these criteria, they are merged.

Transition and acceptance probabilities are estimated from the training sequences. A state, for example, may have been visited n times during training and emitted an 'a' for k of those visits. The probability associated with this state's out-transition labeled 'a' is simply k/n. With a limited training sample, the above criteria for equivalence are rarely met. Consequently, instead of equivalence, Alergia asks whether two states are *compatible*, where compatibility is a probabilistic equivalence. Hoeffding bounds are used to compare inter-state acceptance and transition probabilities:

$$\left|\frac{f_1}{n_1} - \frac{f_2}{n_2}\right| < \sqrt{\frac{1}{2}\log\frac{2}{\alpha}}\left(\frac{1}{\sqrt{n_1}} + \frac{1}{\sqrt{n_2}}\right)$$

Here, n_i is the number of trials and f_i the number of successes particular to the behavior of some state i. Suppose we want to know whether States 1 and 2 have compatible acceptance behavior. Then n_1 is the total number of times any training sequence visited State 1, and f_1 is the number of sequences that terminated at State 1 (the number of times State 1 accepted). The variables n_2 and f_2 stand for the same quantities associated with State 2. The α parameter, controls the certainty of the equivalence judgment. Lower values of α cause more states to be judged equivalent and result in more aggressive generalization. Thus, α is a "knob" which must be set before training can occur. Successful inference depends on choosing an appropriate value.

4.2 Inferring Transducers

In order to conduct grammatical inference effectively, text fragments must be represented as sequences of symbols from an alphabet of manageable size. One possibility would be to regard a field instance as a sequence of ASCII characters, perhaps allowing generalization to exploit some abstract character classes, as in (Goan *et al.*, 1996). This would result in a relatively large alphabet and long sequences, and would probably require a large amount of data to permit effective generalization. For this task, there appears to be more power in structural aspects of entire tokens. Note that the same consideration—limited data—argues against using the literal tokens as alphabet symbols. Also, adopting such a representation would amount to a variation of Rote and would typically result in low recall. Instead, because we want estimates for as many fragments as possible, we want an abstract representation that favors high recall.

A more interesting idea, therefore, is to replace tokens with symbols that correspond to abstract token features, where abstraction is controlled by the structure of the field in question. For example, we would like to transduce the seminar speaker fragment, "`Dr. Koltanowski`", to something like

[tokenDr, token., capitalizedTrue]

i.e., a three-symbol sequence that effectively retains important, high-frequency tokens, but which replaces low-frequency tokens with abstractions that are relevant to the speaker field. Under this scheme, a field instance consisting of five tokens becomes a sequence of five symbols, each symbol the canonical representation of the corresponding token. I will call the procedure that transforms text in this way an *alphabet transducer*.

It is unlikely that any single transducer will be best for all fields, even if it is adapted to a single domain. Consequently, I take a learning approach to constructing field-specific transducers. Ideally, the representation of a token should depend on its position in a fragment, even the grammar state at which it is observed. If our grammar has observed the symbol `Wean` as part of seminar location, we do not want our representation to replace both "`Hall`" and "`5409`" with the symbol `Four-character-token`, since "`Wean 5409`" is a complete location, while "`Wean Hall`" is not.

For simplicity, the approach I adopt assumes that token ordering information can be disregarded in constructing a transducer for a field. The learning problem is shown in Table 4.1. Individual tokens are stripped of their context and labeled only according to

Input:
Positive: All field tokens in corpus
Negative: All non-field tokens in corpus
Features: A set of token-oriented abstract features

Output:
Function mapping a token to its abstract representation

TABLE 4.1: I/O behavior of transducer induction.

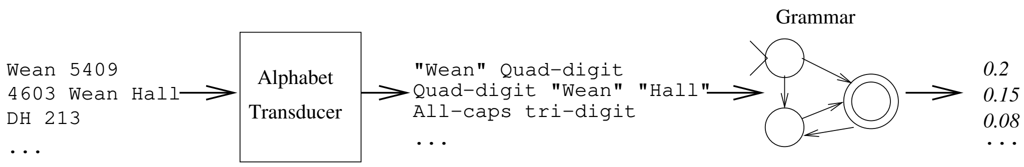

FIGURE 4.6: The pipeline through which raw text fragments are passed to produce structure estimates.

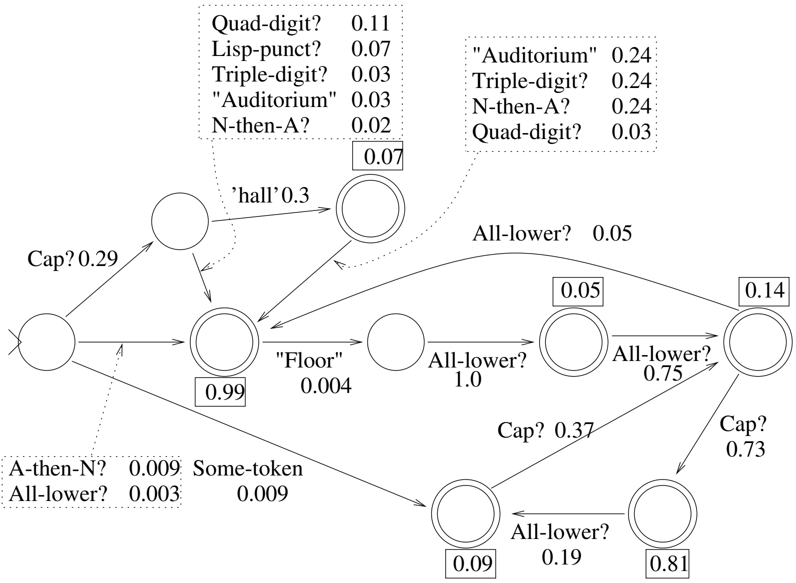

FIGURE 4.7: A small piece of an automaton, used for recognizing seminar locations, learned by Alergia using a decision list created with m-estimates.

If	*Emit*
word = "wean"	`word+wean`
word = "hall"	`word+hall`
triple-digit = true	`triple-digit+true`
quad-digit = true	`quad-digit+true`
capitalized = true	`capitalized+true`
...	

TABLE 4.2: Excerpt from one decision list inferred for the location field.

word	single_digit_p	long_char_p	punctuationp
singletonp	double_char_p	long_digit_p	hybrid_anum_p
doubletonp	double_digit_p	capitalized_p	a_then_num_p
tripletonp	triple_char_p	all_upper_case_p	num_then_a_p
quadrupletonp	triple_digit_p	all_lower_case_p	multi_word_cap_p
longp	quadruple_char_p	numericp	
single_char_p	quadruple_digit_p	sentence_punct_p	

TABLE 4.3: The features used for inferring alphabet transducers.

whether they occur in an instance of the field in question. With the learned transducer, the overall pipeline is illustrated in Figure 4.6. An excerpt from a learned grammar is shown in Figure 4.7. The automaton from which this was taken contained a total of 100 states. Numbers next to emissions are transition probabilities, while those in boxes are acceptance probabilities. Dotted boxes containing multiple emission/probability pairs represent multiple transitions. Combining the two learned components, the transducer and the grammar, yields a function from a raw candidate field instance to an estimate of its structural membership in the target field.

For the learned transducer I use a *decision list* representation. Table 4.2 shows part of a sample decision list, which is a list of pattern/emission rules, for the location field. To transduce a token, we compare it with each pattern in turn until a matching one is found. The token is then replaced with the corresponding symbol. If no matching pattern is found, the token is replaced with the symbol `unknown`. The idea is that more salient patterns will appear higher in the list, so that even if a token matches more than one pattern, it will always be represented in the most useful way. Given the decision list in Table 4.2, for example, the word "Wean" will always cause the symbol `word+wean` to be emitted, even though it also matches the *capitalized = true* pattern. This pattern is reserved for tokens less useful than "Wean" in identifying seminar locations.

A covering algorithm is used to construct decision lists. One input to the learning procedure is a set of features to consider in forming patterns. In the experiments reported

```
1 Function InferATList(docs, field, features)
2    decisionList = the empty list
3    positiveFeatHash = empty hash table
4    anyFeatHash = empty hash table
5    tcount = FieldTokenUncoveredCount(docs, field, decisionList)
6    acount = AnyTokenUncoveredCount(docs, decisionList)
7    While tcount >= MinimumFieldTokensUncovered
8       Set all feature-value entries in hash tables to 0
9       DoPositiveAccounting(docs, field, features, positiveFeatHash)
10      DoAnyAccounting(docs, field, features, anyFeatHash)
11      (feat, value) = FindBestFV(tcount, acount, positiveFeatHash, anyFeatHash)
12      AddToDecisionList(decisionList, feat, value)
13      tcount = FieldTokenUncoveredCount(docs, field, decisionList)
14      acount = AnyTokenUncoveredCount(docs, decisionList)
15   End While
16   Return decisionList
17 End Function
```

TABLE 4.4: The covering procedure used to construct alphabet transducers.

here, I used the 26 features shown in Figure 4.3. Values for all of these features can be readily computed by direct inspection of a token. The word feature returns the literal token, modulo capitalization. Here, longp means longer than four characters in length. Construction of the decision list proceeds greedily. At each step, a feature-value pattern that matches some of the positive tokens is appended to the end of the list, and all matching tokens are removed from the set of positive examples. This process repeats until a stopping criterion is reached, either too few tokens remain in the set of positive examples, or the list has reached a pre-specified length.

Table 4.4 presents pseudocode of the procedure used to build the decision list. The hash tables `positiveFeatHash` (line 3) and `anyFeatHash` (line 4) are used to map feature/value pairs to integer counts—the number of times a feature/value tested true for a field token and overall, respectively. Counts of the number of uncovered field tokens and general tokens remaining are provided by `FieldTokenUncoveredCount()` (lines 5 and 13) and `AnyTokenUncoveredCount()` (lines 6 and 14), respectively. At each iteration of the while-loop (lines 7 through 15) the algorithm scans the document collection, counting the number of times each feature/value test holds for field tokens (`DoPositiveAccounting`, line 9) and, depending on the objective function, for tokens in general (`DoAnyAccounting`, line 10). The choice of which feature/value to install in the list is made by `FindBestFV()` (line 11).

How should we choose the patterns to add to our list? Should we prefer a large list (alphabet) or a small one? Is it important that the patterns used are those that tend to distinguish field tokens from non-field tokens? Or is it sufficient to choose patterns which distribute field tokens evenly? I experiment with three basic approaches:

- **Spread evenly.** Decide how many symbols (k) we want in our alphabet prior to inference of the decision list. At each step i, choose the feature-value test that comes closest to matching $1/(k-i-1)$ of the remaining positive tokens. The set of non-field tokens is disregarded.

- **FOIL gain.** At each step, choose the feature-value pair (call this test ω) that maximizes $f_\omega(\log(f_\omega/n_\omega) - \log(f/n))$, where f is the number of field-instance tokens remaining, n is the total number of tokens remaining, and f_ω and n_ω are the number of field-instance and general tokens, respectively, that match ω. This is similar to choosing the first test of a FOIL rule to distinguish field-instance tokens from general tokens (Quinlan, 1990).

- **M-estimates.** Choose the feature-value pair that maximizes $\frac{f_\omega+m/n}{n_\omega+m}$, for a small value of m ($m = 3$ in these experiments). This rests on the same intuition as for FOIL gain, but uses a different metric.

The effect of the *spread evenly* approach is to sort field tokens into k bins of roughly equal size. Underlying this approach is the hypothesis that it will be possible to infer discriminative grammars even without discriminative symbols—as long as the alphabet is rich enough to provide for interesting multi-token sequences. Note that this approach does not arrange decision-list patterns in order of salience to a field. In contrast, the other two approaches do select patterns that tend to discriminate field tokens from non-field tokens, so the symbols they emit contain more information that those emitted by the first approach. FOIL gain prefers more general patterns at each step than M-estimates and consequently tends to generate smaller transducers.

4.3 Experiments

The same 5-fold experimental framework, with the same partitions, was used in these experiments as in Chapter 3. First, the training set was used to construct an alphabet transducer. Next, the transducer was used to represent field instances as symbol sequences, and Alergia was trained on the resulting sequences. Finally, each of the following extractors was tested on the test set: BayesIDF by itself, the Alergia grammar by itself, and BayesIDF combined with the grammar, as described in this chapter. For succinctness I call this last extractor BayesGI.

I tried five methods for generating transducers:

M-estimates Use m-estimates, as described above, with m set to a small value.

Information gain Use information gain, as described above.

Spread 5 Choose tests that spread field tokens as evenly as possible into five bins.

Spread 10 Like Spread 5, but spread over 10 bins.

Spread 20 Like Spread 5, but spread over 20 bins.

	Approx. 25% recall				*Full recall*			
	Alergia		BayesGI		Alergia		BayesGI	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
CA	—	—	74.8 ± 4.6	25.0	15.8	18.6	46.0	53.3
$\alpha = 0.9$	—	—	67.2 ± 4.7	25.0	16.3	19.2	42.2	49.1
$\alpha = 0.8$	—	—	66.3 ± 4.7	25.0	17.3	20.3	41.1	47.8
$\alpha = 0.5$	—	—	63.9 ± 4.7	25.0	17.2	20.2	41.4	48.2
	BayesIDF				BayesIDF			
	Prec		*Rec*		*Prec*		*Rec*	
	35.6 ± 3.5		25.0		28.8		27.4	

TABLE 4.5: Precision/recall results for Alergia and BayesGI on the *speaker* field, with the alphabet transducer produced using m-estimates, at various settings of Alergia's generalization parameter.

	Approx. 25% recall				*Full recall*			
	Alergia		BayesGI		Alergia		BayesGI	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
CA	99.0 ± 1.1	25.3	99.3 ± 0.9	25.0	42.5	44.4	68.1	66.6
$\alpha = 0.9$	68.3 ± 4.4	25.1	98.6 ± 1.3	25.0	39.8	41.7	59.3	61.0
$\alpha = 0.8$	97.7 ± 1.7	25.2	99.0 ± 1.2	25.0	35.2	36.9	60.4	59.7
$\alpha = 0.5$	34.9 ± 3.2	25.2	99.3 ± 0.9	25.1	27.1	28.3	57.9	57.3
	BayesIDF				BayesIDF			
	Prec		*Rec*		*Prec*		*Rec*	
	97.7 ± 1.7		25.2		57.3		58.8	

TABLE 4.6: Precision/recall results for BayesIDF, Alergia, and BayesGI on the *location* field, using the m-estimates alphabet transducer, at various settings of Alergia's generalization parameter.

For each transducer so constructed, I tried four settings of α, Alergia's generalization parameter: CA, 0.9, 0.8, and 0.5. The CA setting used the canonical acceptor without merging states. Note that, because the transduction step generalizes field instances, CA is not a rote learner. The other settings correspond to increasingly aggressive settings of the generalization parameter; lower settings yield smaller, more general grammars.

Tables 4.5 and 4.6 show the effect of the various settings of α for the *speaker* and *location* fields, respectively. Confidence intervals are at the 95% level. These tables used the m-estimate transducer, because m-estimates yielded the best transducers of the methods we tried, but it is consistent with results for other transducers. The most important conclusion to be drawn from these tables is that BayesIDF benefits a great deal from access to

	Approx. 25% recall				*Full recall*			
	CA		BayesGI		CA		BayesGI	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
M. Est.	—	—	74.8 ± 4.6	25.0	15.8	18.6	46.0	53.3
I. Gain	—	—	63.1 ± 4.7	25.0	5.6	6.6	37.8	44.0
Spread 5	—	—	68.6 ± 4.7	25.0	12.4	14.6	37.7	43.9
Spread 10	—	—	70.7 ± 4.7	25.0	13.4	15.8	37.4	43.2
Spread 20	25.0 ± 2.6	25.2	77.9 ± 4.5	25.0	21.7	25.6	44.7	50.8

TABLE 4.7: Precision results on the *speaker* field for canonical acceptors (with and without BayesIDF) using five different alphabets.

	Approx. 50% recall				*Full recall*			
	CA		BayesGI		CA		BayesGI	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
M. Est.	99.0 ± 1.1	25.3	99.3 ± 0.9	25.0	42.5	44.4	68.1	66.6
I. Gain	41.4 ± 3.6	25.2	99.0 ± 1.2	25.0	30.1	31.5	58.8	61.2
Spread 5	—	—	—	—	6.1	6.4	9.4	9.7
Spread 10	88.0 ± 3.5	24.5	97.3 ± 1.8	25.0	31.7	33.2	45.3	46.6
Spread 20	72.7 ± 4.4	25.0	99.0 ± 1.2	25.0	29.3	30.7	61.7	63.7

TABLE 4.8: Precision results on the *location* field for canonical acceptors (with and without BayesIDF) using five different alphabets.

the structural information a grammar provides. This benefit is realized despite the surprisingly poor performance of the stand-alone grammars, which is due to over-generalization brought on by reliance on positive data alone.

The second general conclusion is that, while the α setting appears to have little effect on the isolated grammar, it does make a difference when this grammar is combined with BayesIDF. In particular, state merging appears to *hurt* performance. Although the differences in precision between the canonical acceptor and the best merged automaton are not statistically significant at the 95% confidence level, this is a consistent effect across all transduction methods and fields. It appears that the abstraction afforded by transduction provides all of the benefit, which state merging diminishes.

What, then, are the factors that influence a good transduction for this problem? Is the size of the resulting alphabet important? Is it important to choose a transduction method that seeks to distinguish field tokens from non-field tokens? Tables 4.7 (precision/recall scores on *speaker*), 4.8 (precision/recall scores on *location*), and 4.9 (alphabet sizes learned by the two "Gain" methods for all four seminar announcement fields) provide some insight. Alphabet (decision list) size is clearly a factor that influences the usefulness of the resulting

	speaker	*location*	*stime*	*etime*
I. Gain	3.2	13.4	7.2	6.6
M. Est.	34.8	48.2	24.8	20.8

TABLE 4.9: Average size of decision lists generated using information gain and m-estimate metrics across the four seminar announcement fields.

	speaker	location	stime	etime
Alergia	17.2	45.1	57.5	46.6
BayesIDF	29.7	61.3	**98.2**	92.3
BayesGI	**52.4**	**71.4**	96.3	89.9

TABLE 4.10: Peak F1 scores for Alergia, BayesIDF, and BayesGI on the seminar announcement fields.

grammar. This result ran counter to my expectations. I was concerned that large alphabets would stand in the way of effective generalization over the essential elements of field structure.

Tables 4.7 and 4.8 do not support a clear preference of m-estimates over Spread 20. From a comparison over all fields, I conclude that the m-estimates method performs slightly better. The difference is small enough in all cases, however, to leave room for doubt. Rather than an abstraction that helps distinguish instances from non-instances, what appears to be important is a large enough alphabet to allow effective representation of the overall structure. As seen in Table 4.9, m-estimates produce on average the largest alphabets.

It is counter-intuitive that the combination of no state merging and large alphabet sizes should yield the best results. This may simply be a task for which it is important to err on the side of specificity. Note that even if sequences of raw field tokens—the most specific representation possible—are used as input for grammatical inference, it is often impossible to construct a grammar that separates field instances from all other fragments. Without attention to context, for example, there is no way to tell whether the fragment "`2:00 pm`" is a seminar start time or the time of some other event associated with a seminar. Abstracting away from the literal tokens exacerbates this problem. The successful transducers, therefore, are conservative, causing any tokens that occur more than a few times to pass through literally (using the word features and selecting abstract representations only for uncommon or generic tokens.

Table 4.10 presents an F1 summary for the four seminar announcement fields (recall that bold face indicates the best peak F1 with high confidence), and Figures 4.8 through 4.11 show the entire precision/recall performance of BayesIDF and Alergia, both alone and combined, on *speaker*, *location*, *stime*, *etime*, respectively. The grammar used to produce

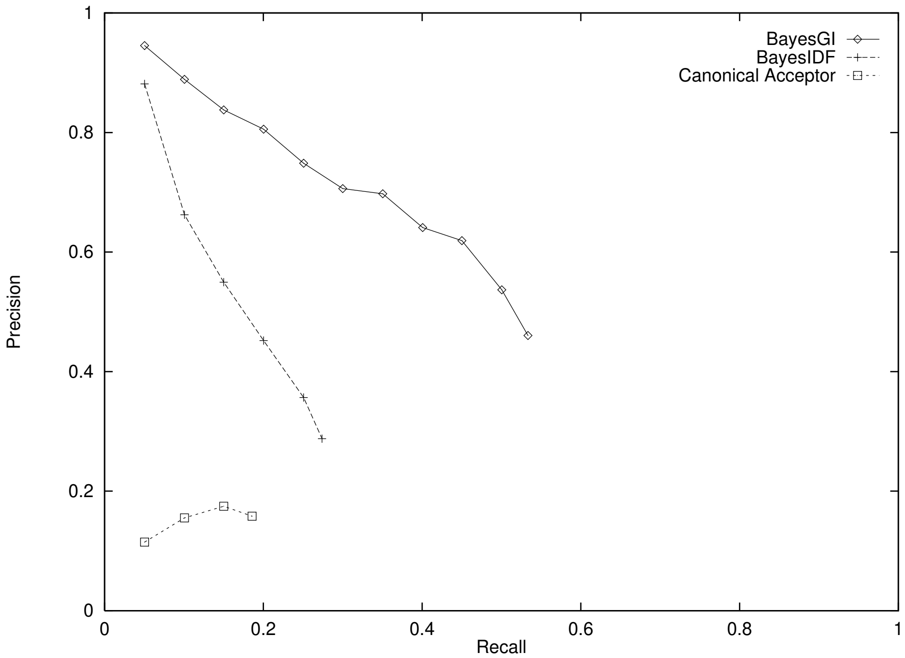

FIGURE 4.8: Precision/recall plot comparing BayesGI, BayesIDF, and the canonical acceptor (CA grammar) on *speaker*.

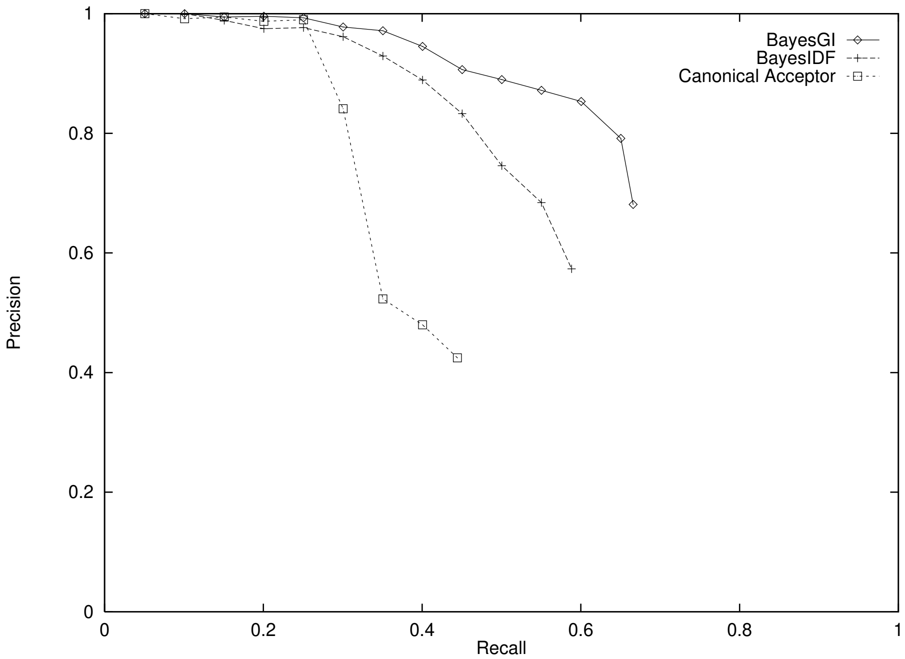

FIGURE 4.9: Precision/recall plot comparing BayesGI, BayesIDF, and the canonical acceptor (CA grammar) on *location*.

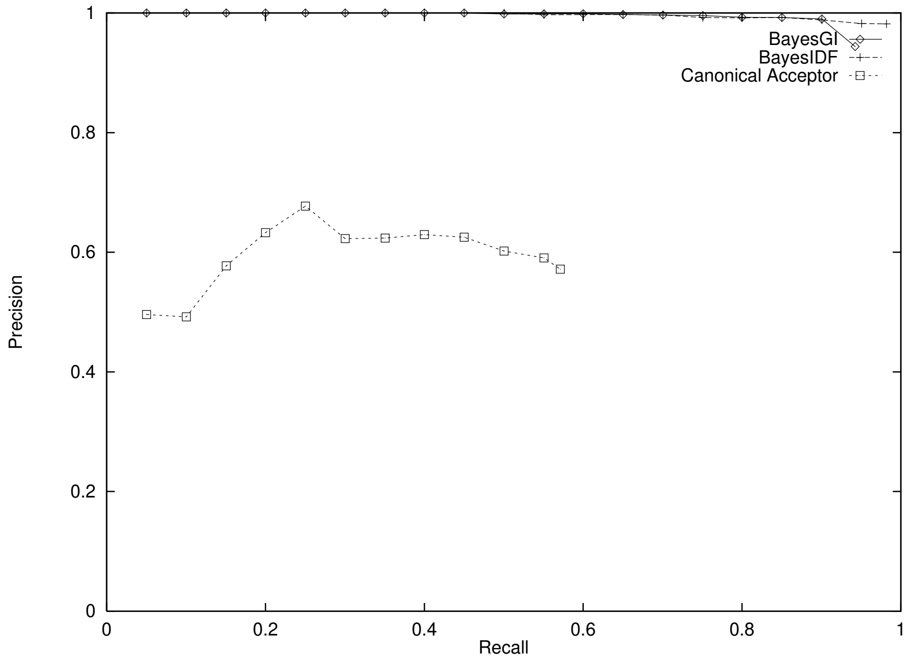

FIGURE 4.10: Precision/recall plot comparing BayesGI, BayesIDF, and the canonical acceptor (CA grammar) on *stime*.

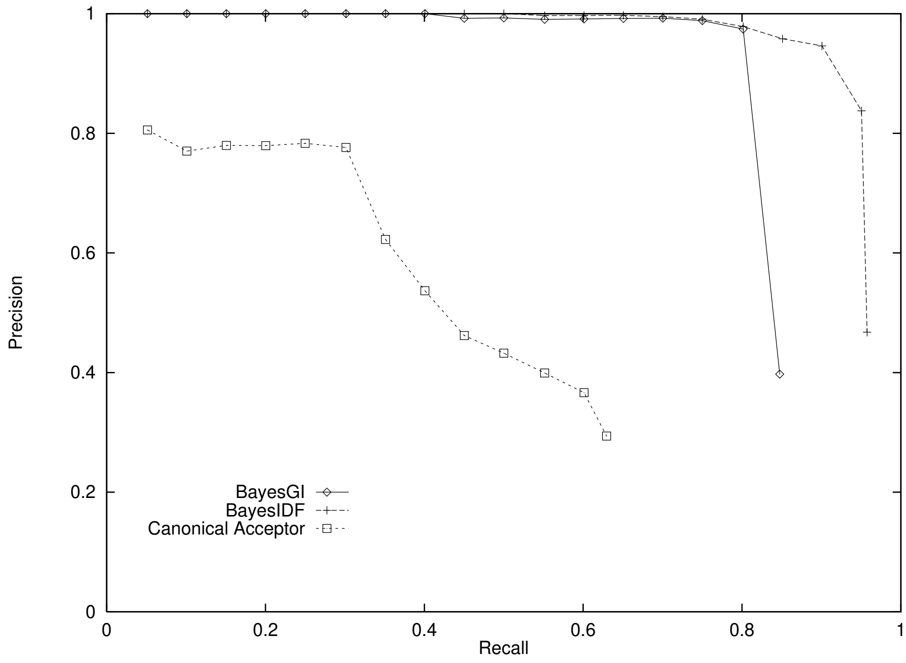

FIGURE 4.11: Precision/recall plot comparing BayesGI, BayesIDF, and the canonical acceptor (CA grammar) on *etime*.

these results was generated on sequences produced by the m-estimate transducer inference method, and the generalization level was set to CA. Improvements due to the combination of methods are obvious for the two fields on which BayesIDF stands to improve, *speaker* and *location*. BayesGI performs slightly worse than BayesIDF on the two time fields. The drop in performance, however, occurs only at the high-recall end of the curve.

4.4 Discussion

Why does the multistrategy combination of BayesIDF and the token grammar provide such benefit? What are the general lessons to be drawn from these experiments? It seems fairly obvious that the fact that the two constituent learners attend to different aspects of the problem is the source of the performance boost. If the two learners had the same behavior—accepting and rejecting the same text fragments—we could not expect to see improvement by combining them. In other words, the strength of their combination depends on their diversity in terms of representation and bias. On a hard problem, any individual learners experiences parts of the space which are difficult for its particular bias. It has been suggested that a learner's strengths in some parts of its hypothesis space necessarily entail weaknesses in others (Schaffer, 1994). The hope of multistrategy learning is that the weaknesses of one learner will be covered, partially, by the strengths of another.

While the positive results obtained for the method described here are gratifying, this method leaves something to be desired. Since most of the benefit seems to come from identifying the most salient structural aspects of tokens, it would be nice to have a more principled approach to finding and representing this structure. The current methods are coarse. Treating all the field tokens as elements of a large set for the purposes of training a transducer, ignoring co-occurrence and ordering information, while effective, appears to neglect useful information. Furthermore, the decision-list formalism forces us to choose *one* feature of a token with which to represent it. This, too, is a waste of available information. Ultimately, we might want to discard the transduction step altogether, to design something like a grammatical inference algorithm that can work with sequences of *feature vectors*. These considerations are a subject for future work.

Chapter 5

Relational Learning for Information Extraction

> Like the algorithm presented in Chapter 4 that learns alphabet transducers, symbolic rule-learning algorithms are also well situated to make use of token features. The end target of that algorithm, however, was not the classification of fragments. In contrast, this chapter presents a symbolic rule learner that searches for extraction patterns based on token features directly. It describes SRV, a relational learner for information extraction. Relational learning refers to a class of symbolic learners that search a space of relations between examples. SRV's relational component is designed to allow it to explore arbitrary amounts of field context. Experiments in three domains demonstrate both its versatility in exploiting domain-specific information, by means of hand-crafted token features, and its comparative superiority in precision and recall over the other three learners.

The results presented in Chapter 4 are sufficiently convincing to establish that useful extractors can be based on simple token features. However, the solution presented there leaves a few things to be desired:

- Although the combination of BayesIDF and grammatical inference works, the way in which they are combined is somewhat arbitrary. The structural estimate returned by grammatical inference is dropped in a naive way into BayesIDF's already naive estimate. There is no guarantee that this assigns the structural estimate its proper weight.

- Grammatical inference must express its patterns in terms of all tokens in a field. It may be the case, however, that only certain aspects of some of the tokens are important. If this is so, then the requirement that all tokens be accounted for could hamper generalization.

- Training occurs in two separate phases, inferring a transducer and building a grammar. Feature-value selection is decoupled from the process of building a classifier to recognize field instances.
- Although BayesGI, through BayesIDF, has a notion of field context, its grammatical inference component does not. Thus, any interesting structural patterns in the tokens immediately surrounding a field's instances are lost.

In summary, it would be desirable to have a learner that can apply token features more flexibly. The application of any particular feature should be based directly on its usefulness in distinguishing field instances from text in general. And this learner should not be limited to in-field tokens in its use of such features, but should have the ability to apply them to contextual tokens as well.

5.1 SRV

In order to meet this requirement, this chapter considers the family of *symbolic* inductive learners, which includes decision tree learners, such as C4.5 (Quinlan, 1993), covering algorithms, such as AQ (Michalski, 1983) and CN2 (Clark and Boswell, 1991), and inductive logic programming or *relational* learners, such as FOIL (Quinlan, 1990). Not only do learners in this class hold the promise that abstract features of a domain can be used effectively and directly, but the structure of the classifier they produce is also attractive for the information extraction problem. Their bias is divide and conquer. The learned classifier is a set of rules which match sub-patterns in a class, and which are disjunctively combined to make predictions. This seems to fit many information extraction tasks well. Often, we can identify multiple distinct patterns for a field, any one of which can indicate the presence of an instance by itself. This section describes SRV, a relational learner for information extraction.

A symbolic learner typically takes two kinds of input: a representation language and a set of examples to be used in training. These examples are divided into n classes. The goal of the learner is to produce a set of logical rules (or their functional equivalent) to classify each novel example into one of these classes.

In propositional learning examples are defined in terms of *features*, which are functions mapping examples to typically discrete values. If each example is a day's worth of weather, for instance, a possible feature is *rainy*, a function, the range of which is days and the domain of which is the set $\{yes, no\}$. Given such a feature, we can express a simple fact about a particular day:

$$rainy(today) = yes$$

In contrast, in relational learning examples are defined in terms of *predicates*, which are relations. To stick with the weather example, in addition to unary *predicates* like *Rainy*, our example space may be defined in terms of binary predicates such as

Fragment:	`... will meet in` $\boxed{\texttt{Baker Hall 300}}$`, at 3:30 ...`
Positive example:	`Baker Hall 300`

Negative examples:	`will meet`
	`will meet in`
	...
	`meet in`
	`meet in Baker`
	...
	`Baker Hall 300,`
	`Baker Hall 300, at`
	...
	Hall 300
	...

FIGURE 5.1: A text fragment and some of the examples it generates—one positive example, and many negative ones.

Followed(*day1*, *day2*)

to express a succession of days, or the ternary predicate

TempExtremes(*Day*, *Low*, *High*)

to express the range of temperatures seen on a particular day. In principle, there is no limit to the arity of predicates that can be used to describe examples.

While in propositional learning examples are so many separate entities, the predicates of a particular relational learning problem implicitly or explicitly relate examples to each other. Relational learners are designed to recognize and exploit such inter-example structure. This facility seems appropriate for the information extraction problem, where examples (text fragments) are embedded in a larger structure and implicitly related to the text that surrounds them in a number of ways. This section describes one way such natural relational structure of the information extraction problem can be exploited.

5.1.1 Example Space

Like other covering algorithms, SRV requires that the learning problem consist of a set of examples, some positive and some negative—the *example space*. For SRV, examples are text fragments. Instances of the field SRV is learning to extract are positive examples. As a preprocessing step SRV scans the training corpus to find two numbers, *min* and *max*, the number of tokens in the smallest and largest field instances, respectively. Subsequently, during training and testing, SRV regards as a negative example *any fragment having at least* min *tokens and no more than* max *tokens that is not a field instance*. All such fragments are counted and examined as part of training and testing. Table 5.1 shows the generation of examples from a hypothetical fragment. Note that examples overlap densely; a positive example typically shares tokens and context with many negative examples. How many negative examples are are defined for a learning problem depends in large part on the range of field instance sizes observed in the training corpus.

In discussing the learners described in previous chapters, I have been able to skirt the problem of defining the space of negative examples explicitly. In the case of BayesIDF the idea of a negative example is mainly implicit; the algorithm works with a set of term-frequency tables. Rote and Alergia, on the other hand, only work with positive examples—the set of field instances.[1] By adopting the paradigm of set-covering algorithms, however, we are forced to confront this problem. Even with the fragment-size limitation, the set of negative examples for any particular learning problem that SRV faces is typically several orders of magnitude larger than the set of positive examples. Much of the challenge of implementing a learner like SRV lies in developing strategies to cope with this large negative set.

5.1.2 Features

SRV's induction procedure is based on the notion of *features*, which are functions over individual tokens. Features come in two basic types. A *simple* feature is a function that maps a token to an arbitrary value, which is categorical and usually Boolean.[2] An example of this kind of feature is capitalized, which takes the value *true* for any token beginning with a capital letter and *false* otherwise. A *relational* feature, on the other hand, is a function that maps a token to another token in the same document. An example is next_token, which returns the token immediately following its argument, or *undefined* if its argument is the last token in the document.

Note that relational features are what gives SRV its relational character. Each such feature encodes one-half of a binary relation. For example, the binary relation $Succeeds(token1, token2)$ is equivalent to two statements using relational features:

$$next\_token(token1) = token2$$

and

$$prev\_token(token2) = token1$$

In SRV, relational features are used instead of predicates for the sake of convenience and efficiency. Not only are they similar in form to simple features, but they also limit SRV's searching ability in a way that is reasonable for the information extraction problem.

[1]Of course, Rote's statistics count false matches in the training set. The negative examples it must examine in this way, however, is determined and strongly constrained by the contents of its dictionary.

[2]Simple features may also be set-valued, i.e., they may return a set of values. Imagine, for example, a (probably useless) feature that returns all the letters used in a token—the value of $letters(\texttt{Dog})$ would be $\{`d', `o', `g'\}$. SRV is designed to expect such features, but I have only implemented one, wn_word, which is discussed later in the chapter. For simplicity, most of the discussion in this chapter assumes categorical simple features.

Features, as they are understood by SRV, differ in one respect from the features of a conventional covering algorithm, such as CN2. In such an algorithm features describe aspects of the examples themselves. If the learning problem is to identify storm clouds, features such as is_white, distinct_border, and size might be used—all features of clouds. In contrast, SRV's features describe aspects of an example's *components*, the tokens that make up the fragment. The fact is, while it might be desirable to invent features of multi-token fragments, it is difficult to come up with a satisfactory set of such features. One can readily speak of the length of a fragment and its position within a document, but this is only a small portion of the information that might be exploited.

In contrast, token features are easy to define—a fact that facilitates adaptation of an algorithm to new domains and new sources of information. Given a token drawn from a document, a number of obvious feature types suggest themselves, such as length (e.g., single_character_word), character type (e.g., numeric), typography (e.g., capitalized), part of speech (e.g., verb), and lexical meaning (e.g., geographical_place). Similarly, in addition to relational features that encode token adjacency, it is straightforward to encode structural aspects of text like linguistic syntax (e.g., subject_verb) in a form that SRV is able to exploit.

5.1.3 Rule Construction

SRV constructs rules "top-down," starting with null rules that cover the entire set of examples—all negative examples and any positive examples not covered by already induced rules—and adding literals greedily, attempting thereby to cover as many positive examples as possible while weeding out covered negative examples. In the discussion that follows, I first present example literals in the SRV-specific form used throughout the rest of the chapter, then give translations in both first-order logic and English. Note that all SRV literals implicitly refer to fragments. In translations I use F to stand for a matching fragments Predicates can be instantiated from any of the five following templates.

- length(Relop, N): Here Relop is one of $\{<, >, =\}$ and N is an integer. This literal asserts that the number of tokens in a fragment is less than, greater than, or equal to some integer. For example, the literal

  length($<$, 3)

  has the logical meaning

  $$length(F) < 3$$

  where the function *length* returns the number of tokens in F. In other words, it posits that only fragments one or two tokens in length should be considered.

- some(Var, Path, Feat, Value): Var is a variable, Path is a list of relational features, Feat is a simple features and Value is a legal value of Feat. This is a feature-value test for some token in the sequence, which is bound to the variable Var. Each distinct variable in a rule must bind to a distinct token in a matching fragment. In logical terms, if SRV has previously introduced the variable A and now, by means of a new some-literal, introduces B, there is an implicit assertion of inequality between A and B:

$$\exists A \in F.\exists B \in F.A \neq B \wedge \cdots$$

An example of the some-literal might be

some(?A, [ ], capitalized, true)

which can be rendered in first-order logic as

$$\exists A \in F.\mathit{capitalized}(A) = \mathit{true}$$

and in English as

"F contains some token that is capitalized (call it '?A')."

The Path argument, which is empty in this example, is used to exploit relational structure in the domain (see below). Note that every some-literal in the same rule using the same token variable is required to refer to the same token. In logical terms, each such assertion falls into the scope of the same existential quantifier. Thus, if after making the example some-assertion above, the literal

some(?A, [ ], single_char, false)

is added to the rule, the following assertion holds:

$$\exists A \in F.\mathit{capitalized}(A) = \mathit{true} \wedge \mathit{single\_char}(A) = \mathit{false}$$

- every(Feat, Value): Feat is a simple feature and Value is a legal value of Feat. Every token in a fragment passes some feature-value test. For example,

  every(numericp, false)

  has the logical meaning

  $$\forall A \in F.\mathit{numericp}(A) = \mathit{false}$$

  and the English meaning

  "Every token in F is non-numeric."

- position(Var, From, Relop, N): Var is a variable, From is one of {fromfirst, fromlast}, Relop is one of $\{<, >, =\}$, and N is an integer. This constrains the position of a token bound by a some-literal in the current rule. The position is specified relative to the beginning or end of the sequence. Example:

  position(?A, fromfirst, $<$, 2)

In logic this can be rendered as

$$\exists A \in F.\textit{dist\_from\_first}(A, F) < 2 \wedge \cdots$$

where *dist_from_first* is a function counting the number of positions A is from the start of the fragment (0 if the first token) and the ellipsis ($\cdots$) stands for whatever other assertions have been made about A (one must already have been made). In English this literal can be translated as:

> "The token bound to ?A is either the first or second token in the fragment."

- relpos(Var1, Var2, Relop, N): Var1 and Var2 are variables, Relop is one of $\{<, >, =\}$, and N is an integer. This constrains the ordering and distance between two tokens bound by distinct variables in the current rule. Example:

  relpos(?A, ?B, =, 1)

  In logic:

  $$\exists A \in F.\exists B \in F.\textit{position\_diff}(B, A) = 1 \wedge \cdots$$

  where *position_diff* is a function returning the difference of the position indexes of its two arguments, and the ellipsis again stands for previously made assertions about A and B. In English:

  > "The token bound to ?A immediately precedes the token bound to ?B."

Relational features are used only in the Path argument to the some predicate. This argument can be empty, in which case the some literal is asserting a feature-value test for a token occurring within a field, or it can be a list of relational features, in which case it is positing both a relationship between a field token and some nearby token, as well as a feature-value for the other token. For instance, the assertion

some(?A, [ ], all_lower_case, true)

represents the logical statement

$$\exists A \in F.\textit{all\_lower\_case}(A) = \textit{true}$$

and can be translated in English as

> "the fragment contains some token (?A) that consists entirely of lower-case letters."

In contrast, the assertion

some(?A, [prev_token prev_token], all_lower_case, true) corresponds to the logical statement

$$\exists A \in F.\mathit{all\_lower\_case}(\mathit{prev\_token}(\mathit{prev\_token}(A))) = \mathit{true}$$

and can be rendered in English as

> "There is some token in the fragment preceded by a lower-case token two tokens back."

There is no limit to the number of relational features the learner can string together in this way. Thus, it is possible in principle for the learner to exploit relations between tokens quite distant from each other. And the same set of simple features which are used to describe tokens within a fragment are available to describe extra-fragment tokens.

The most important function of the path argument is to allow the learner to expand its consideration of relevant field structure to the text outside the field instance. These relational paths are powerful, but also potentially very costly in terms of search effort. SRV limits the cost heuristically. At all times during rule construction SRV maintains a set of paths (strings of relational features) which are candidates for use in a new some-literal. Before the construction of each rule, this set initially contains only the null-path and all paths of length one (e.g., [prev_token]). Whenever SRV actually uses a path by adding a some-literal containing it to the rule under construction, it adds to this candidate set all paths created by appending all single relational feature to the path it used. In effect, SRV grows its consideration of surrounding context outward. The rate of growth is governed by how useful it has already found context to be, and how much use it is able to make of information contained within positive examples.

When deciding which literal to add to the rule under construction, SRV uses the same gain metric as FOIL. In describing FOIL (Quinlan, 1990), Quinlan defines a function which characterizes the information contained in the ratio of positive to negative examples in a set of examples:

$$I(S) = -\log_2(P(S)/(P(S) + N(S)))$$

where P(S) and N(S) are the number of positive and negative examples, respectively, in the set S. Suppose S corresponds to a set of examples covered by some partially formed rule, and let S_A stand for the subset of S covered by the rule when the literal A is added. Then the gain metric used in SRV and FOIL is:

$$\mathit{Gain}(A) = P(S_A)(I(S) - I(S_A))$$

In order to calculate the gain of candidate literals, the entire set of examples matching the current rule must be examined. In practice, this entails loading and scanning each document in the training set. A hash table is used to account for candidate literals; with each such literal is associated a record holding two numbers, the number of positive and negative examples the literal matches. After processing the training documents and tabulating these numbers for each candidate literal, SRV scans the hash table to select one literal maximizing the gain criterion.

Construction of a rule stops if either of two conditions holds:

```
<0.26.4.95.11.09.31.hf08+@andrew.cmu.edu.0>
Type:     cmu.andrew.academic.bio
Topic:    "MHC Class II: A Target for Specific
          Immunomodulation of the Immune Response"
Dates:    3-May-95
Time:     3:30 PM
Place:    Mellon Institute Conference Room
```

TABLE 5.1: An excerpt from the header of a seminar announcement.

- The rule matches only positive examples (it achieves *purity*).
- No literal exists for which the *Gain* function returns a positive value.

Often, however, especially after easy patterns have been captured by earlier rules in a training session, a rule learner may induce rules that cover very few positive examples. Not only is the probable lack of generality of such rules undesirable, but they consume search effort without advancing the learner's cause much: If only a single positive example is covered by a very large rule and 100 examples remain to be accounted for, then we can expect the system to spend its effort on 99 more such large, low-coverage rules. In order to avoid this situation, SRV discards any candidate literal that covers fewer than five examples. This number is arbitrary. Although I have not experimented with different settings, it is a parameter which can be changed.

5.1.4 An Example

Consider the problem of identifying the start time of a seminar from an announcement. This turns out to be an easy problem for both SRV and, as we have seen in Chapter 3, BayesIDF. In contrast with BayesIDF, however, SRV forms hypotheses that are at least partially intelligible to humans. In this section I follow the growth of one rule that achieves unusually high coverage on the training set.

In the trace to follow SRV is concerned with learning to extract instance of *stime* from seminar announcements. In the training set there are 161 seminar announcements, in which *stime* is instantiated a total of 325 times—325 positive examples. Instances range from one to six tokens in length. The total number of negative examples in the initial pool is 380,384. Table 5.1 shows an excerpt from the training set containing an instance of *stime* (“`3:30 PM`”).

The first literal asserted by SRV is intuitively appealing:

some(?A, [ ], single_digit_p, true)
Positive: 252
Negative: 26,405
Gain: 874.2

Literals considered: 52,138

SRV asserts that start time fragments contain at least one token that consists of a single digit. The variable '?A' will stand for this token here and in the rest of the rule.

The number of positive and negative examples matched by this proto-rule are also shown. Note that although the rule now covers 252 positive examples out of an initial 325, it still covers a large number of negative examples, as well. To see what these false positives look like, consider Figure 5.1. Only a single start time is instantiated here ("`3:30 PM`"), but there are many fragments that match the current rule. The first line (the message ID) contains a number of singleton digits, so that, for example, the fragments "`26.4.`" and "`.4.95.`" match the rule as it now stands.[3] Remember also that there is an implicit range of candidate fragment lengths, which depends on the observed lengths of training instances (1 to 6 in this particular example). The entire message ID line, for example, is not counted as a negative example.

A number of false positives share tokens with the single true positive example in the excerpt. In general, we can distinguish between two closely related problems an extraction system must solve: locating field instances and identifying boundaries. Sometimes a single literal is enough to exclude all text not in the immediate proximity of some field instances. Even in such cases, however, additional literals are required to exclude all but one of the cloud of overlapping fragments that surround a field instance. In our example, such matching fragments include "`:  3:`" and "`3:30 PM Place`".

The next literal fixes the position of the singleton digit. It must occur at the beginning of the fragment:

some(?A, [ ], single_digit_p, true),
position(?A, fromfirst, $<$, 1)
Positive: 252
Negative: 7649
Gain: 442.1
Literals considered: 1569

Not surprisingly, this literal does not sacrifice any positive examples, but it filters out a large number of negative examples. Now, although the false positive "`3:30 PM Place`" still matches, the fragment "`:        3:`" no longer does.

A new variable is introduced by the third literal, and a relational feature is used:

some(?A, [ ], single_digit_p, true),
position(?A, fromfirst, $<$, 1),
some(?B, [prev_token], word, ":")
Positive: 244
Negative: 2374
Gain: 377.5
Literals considered: 2031

[3]Note, however, that a token such as "`95`" will not bind to '?A'. Thus, there are only three tokens in the message ID that are responsible for false positives.

The word feature returns the original token with all upper-case characters converted to lower case. This literal says, "the fragment contains some token (other than the one bound to '?A') that is preceded by a colon." Note that although the '3' in the start time in Figure 5.1 is also preceded by a colon, this literal does not refer to it. Recall that distinct variables must bind to distinct tokens. Because the '3' is bound to the '?A' variable, '?B' must bind to a different token—the '30' token in this case.

After three literals the set of matching negative examples has been vastly reduced. The only false positives in Figure 5.1 are those associated with the start time and sharing the same initial token ('3'). To reach the current, relatively small number of false positives, we have had to sacrifice eight of the positive examples matched by the initial literal. These are presumably single-token start times, as in, "The seminar will be held at 1." Interestingly, although only two tokens are bound, the three literals together require that a matching fragment contain at least three tokens. Because the singleton digit must be the first token, the colon that precedes the token bound to '?B' must also be part of the start time. Also, all remaining negative examples in the excerpt in Figure 5.1 are those beginning with the fragment "`3:30`".

The next literal places an upper bound on the length of matching fragments:

> some(?A, [ ], single_digit_p, true),
> position(?A, fromfirst, <, 1),
> some(?B, [prev_token], word, ":"),
> length(<, 5)
> *Positive:* 220
> *Negative:* 585
> *Gain:* 341.4
> *Literals considered:* 1074

Following this literal, only fragments containing three or four tokens are considered.

The fifth literal illustrates the growth of relational paths:

> some(?A, [ ], single_digit_p, true),
> position(?A, fromfirst, <, 1),
> some(?B, [prev_token], word, ":"),
> length(<, 5),
> some(?A, [prev_token, prev_token], quadrupletonp, true)
> *Positive:* 163
> *Negative:* 197
> *Gain:* 118.7
> *Literals considered:* 1055

This literal says, "the token two tokens before ?A (the singleton digit that starts the fragment) is exactly four characters long." Because begins rule construction only considering relational paths of no more than length one, and extending the set of candidates only when relational paths are actually used, the path [prev_token, prev_token] was not available when construction began on this rule. Only after the third literal, which used the relational feature prev_token, was this path available. And following the use of this path, all three-feature paths starting with [prev_token prev_token] are also candidates. This literal appears to generalize two common labels used to introduce start times in the collection, "`Time:`", as in Figure 5.1, and "`When:`".

The sixth literal posits an additional token, thereby restricting attention to fragments of length four:

> some(?A, [ ], single_digit_p, true),
> position(?A, fromfirst, <, 1),
> some(?B, [prev_token], word, ":"),
> length(<, 5),
> some(?A, [prev_token prev_token], quadrupletonp, true),
> some(?C, [ ], doubletonp, true)
> *Positive:* 88
> *Negative:* 2
> *Gain:* 97.7

The meaning of this literal is, "the fragment contains some token '?C' containing exactly two characters." Because the only other doubleton token in the start time in Figure 5.1 is already bound to ?B, the variable introduced by this literal ('?C') can only bind to 'PM'.

Two false positives remain to be filtered out. Fortunately, a literal is available to remove them without sacrificing any field instances:

> some(?A, [ ], single_digit_p, true),
> position(?A, fromfirst, <, 1),
> some(?B, [prev_token], word, ":"),
> length(<, 5),
> some(?A, [prev_token, prev_token], quadrupletonp, true),
> some(?C, [ ], doubletonp, true),
> some(?A, [prev_token, prev_token, prev_token], all_lower_case, false)
> *Positive:* 88
> *Negative:* 0
> *Gain:* 2.9
> *Literals considered:* 998

This literal means, "the token three tokens before '?A' (the first token in the fragment) is not a token consisting entirely of lower-case alphabetic characters."

At this point construction of the rule halts. No further improvements are possible, because purity has been achieved—no negative examples are matched. This rule can be rendered in first-order logic as:

```
94 Time:      [3:00 PM] PostedBy: valdes
92 Time:      [9:00 AM] - 12:
92 Time:      [5:00 PM] PostedBy: Saul
92 Time:      [7:00 PM] PostedBy: Career
, 1995; [3:30 pm] - 5:
DOHERTY HALL 1212                              [7:00 PM] ***
```

FIGURE 5.2: Some start times that match the learned rule (carriage returns removed but other whitespace preserved).

$$
\begin{aligned}
\mathit{stime}(F) \;\leftarrow\;& \mathit{length}(F) < 5 \\
\wedge\;& (\exists A, B, C \in F. A \neq B \neq C \\
& \wedge \mathit{single\_digit\_p}(A) = \mathit{true} \\
& \wedge \mathit{quadrupletonp}(\mathit{prev\_token}(\mathit{prev\_token}(A))) = \mathit{true} \\
& \wedge \mathit{all\_lower\_case}(\mathit{prev\_token}(\mathit{prev\_token}(\mathit{prev\_token}(A)))) = \mathit{false} \\
& \wedge \mathit{dist\_from\_first}(A, F) < 1 \\
& \wedge \mathit{word}(\mathit{prev\_token}(B)) = \text{`:'} \\
& \wedge \mathit{doubletonp}(C) = \mathit{true})
\end{aligned}
$$

The rule is stored as part of the output hypothesis, all field instances it matches are removed from further consideration, and construction of a new rule is begun.

SRV has successfully recognized one particularly common pattern of start time: fully specified times ending with the token ‘PM’ or ‘AM’. Figure 5.2 shows a few of the 88 matching examples from this particular training session. In order to distinguish seminar start times from other times listed in the announcements, SRV has relied on the text which precedes instances: Start times are preceded by a token that does not consist of two alphabetic characters, and the token preceding this token contains exactly four characters. It probably would not occur to a human being to express rules in exactly this way. Indeed, although most of the fragments matched by this rule are preceded by the two-token label “`Time:`”, Table 5.2 shows two field instances for which this is not true. Nevertheless, these two fragments are not singular; there are other cases in the set of matching examples where a start time is preceded by a year or a location containing the word “hall.” Thus, without encompassing any kind of semantic understanding, SRV has exploited domain regularities that enable it to perform quite effectively.

5.1.5 Rule Accuracy Estimation

The final training step is rule validation. A randomly selected portion (one-third, in this case) of the training data is set aside for validation prior to training. For each rule learned during the training step, the number of matches and correct predictions on the validation set is tabulated. These numbers are used subsequently to estimate a rule's accuracy.

In order to get as much out of the training data as possible, it is randomly partitioned into three sets of equal size. Each of these sets is then used as the validation set for rules learned on the remaining two-thirds of the data. This yields three independently trained and validated rule sets, which are then concatenated to form the final classifier.

At testing time, associated with each rule are two counts: the number of times it matched any fragment in the validation set, and the number of times it *correctly* matched. The rule's confidence is an m-estimate (Cestnik, 1990), based on these two numbers, of the probability that the rule is correct, given that it matches a fragment.[4] All candidate fragments in a test document are compared with all rules to look for a match. If at least one matching rule is found, SRV predicts that the fragment is a field instance and returns a confidence with this prediction. This confidence is a combination of the confidences of *all* matching rules. Suppose C is this set of confidence scores. The combined confidence is:

$$C_{combined} = 1 - \prod_{c \in C} (1 - c)$$

In other words, the rules are treated as independent predictors. Although this independence assumptions is not justified by the training regime, I have found that this approach works better than only taking the largest confidence in C.

5.1.6 Implementation

Table 5.2 presents SRV's function for growing a single rule. Starting with an empty rule (line 2), SRV iteratively adds the single literal that maximizes its gain metric. Most of this rule growth occurs in the `While`-loop of the function `GrowRule` (lines 13 through 23). For reasons that will be discussed in this section, however, the function called to find the first literal in the rule (`FirstLiteral`, line 8), differs from the main function for finding literals (`NextLiteral`, line 14). Before proceeding with a discussion of the search for literals, note how the set of candidate relational paths is handled in `GrowRule`. Initialized to contain all paths of length zero or one (lines 3 through 6), it is augmented any time a some-literal is added to the rule (lines 11 and 21). The Function `AugmentCandidatePaths` iteratively appends each relational feature in `relational_feats` to each path in `candidate_paths`; any resulting path not already in the set `candidate_paths` is added to it.

In order to perform top-down rule induction for information extraction, attention to efficiency is important. A naive search for an individual literal would involve evaluating

[4] I set m to 3 in experiments involving SRV.

```
Function GrowRule(fieldname, docs, simple_feats, relational_feats)
   rule = empty rule
   candidate_paths = [ the empty path ]  /* Singleton list containing empty path */
   For feat in relational_feats
      AddToList(MakeSingletonPath(feat), candidate_paths)
   End For

   literal = FirstLiteral(fieldname, docs, simple_feats, candidate_paths)
   AddToList(literal, rule)
   path = PathOfSomeLiteral(literal)
   candidate_paths = AugmentCandidatePaths(candidate_paths, path)

   While MatchesNegativeExamples(fieldname, docs, rule)
      literal = NextLiteral(rule, fieldname, docs, simple_feats, candidate_paths)
      If literal = null        /* Failed to find a literal */
         Exit While
      End If
      AddToList(literal, rule)
      If LiteralType(literal) = some
         path = PathOfSomeLiteral(literal)
         candidate_paths = AugmentCandidatePaths(candidate_paths, path)
      End If
   End While
   Return rule
End Function
```

TABLE 5.2: SRV's rule-growing algorithm in pseudocode.

the effect of every candidate literal on every legal fragment. If we are at the beginning of search (i.e., no literals asserted and no positive examples covered), this entails counting each one of approximately $n(\mathit{max} - \mathit{min})$ fragments in a single file, where n is the number of tokens the file contains, and *min* and *max* are the minimum and maximum legal fragment sizes, respectively.

SRV embodies a number of implementation strategies that make search through such a large space efficient. A critical feature of this problem is that examples exist in implicit form in documents on disk. Moreover, because they overlap, this is their most efficient representation; enumerating examples and storing them as a group in main memory is not feasible with typical memory sizes. Consequently, the primary consideration in an efficient implementation is minimizing file I/O.

The core of any greedy general-to-specific symbolic learner is a linear scan of examples, and for each example, a linear scan of available assertions. If x is the number of examples and a is the number of assertions, this means that $O(ax)$ work is required for each assertion added to a rule. In general, whether to scan examples or assertions in the inner loop may be of little consequence; for information extraction, however, it is important that examples be scanned in the outer, assertions in the inner loop. Because scanning examples entails fetching data from disk, we want to perform it only once per *added* assertion, rather than once per *candidate* assertion.

When analyzing the training data to determine which literal to add to a rule, SRV

```
1 Function NextLiteral(rule, fieldname, docs, simple_feats, candidate_paths)
2    Variable positive_hash,     /* Tables mapping to integers ... */
3             negative_hash      /* ... all entries initially zero */
4    vars = VariablesUsed(rule)
5    newvar = variable not in vars
6    For doc in docs
7       bound = Table mapping tokens in doc to vars in rule
8       For fragment in MatchingFragments(doc, rule, bound)
9          If IsFieldInstance(fieldname, doc, fragment)
10            hash = positive_hash
11         Else
12            hash = negative_hash
13         End If
14         SearchForLengthLits(fragment, hash)
15         For var in vars
16            SearchForSomeSpecializations(fragment, bound, simple_feats,
17                                         candidate_paths, hash)
18         End For
19         SearchForNewSomeLits(fragment, newvar, bound, simple_feats,
20                              candidate_paths, hash)
21         SearchForEveryLits(fragment, simple_feats, hash)
22         For var in vars
23            SearchForPositionLits(var, fragment, bound, hash)
24         End For
25         For var1 in vars
26            For var2 in vars
27               If Not var1 = var2
28                  SearchForRelposLits(fragment, bound, hash)
29               End If
30            End For
31         End For
32      End For /* matching fragments */
33    End For /* documents */
34    Return FindBestLiteral(positive_hash, negative_hash)
35 End Function
```

TABLE 5.3: SRV's procedure for finding all but the first literal in a rule.

performs its analysis in this order:

1. Each training document is loaded.

2. Each legal fragment in a document is analyzed.

3. The appropriate count (*positive* or *negative*, depending on the fragment) is incremented for each candidate literal that matches this fragment.

Table 5.3 presents pseudocode for the function `NextLiteral`, which implements this strategy. Enumerating fragments (Step 2 above) is a computationally intensive process. To make it more efficient, SRV exploits the overlap between examples by pre-calculating token-specific information in Step 1. For example, during Step 1 SRV calculates and caches the values of all features for every token in a document. Once this is done, every token is annotated with the variables in the current rule to which it can be bound (line 7 in Table 5.3). Using these annotations, it is possible to screen out many fragments quickly and avoid the expensive process of unification. For example, if a fragment has no token that binds to the variable '?A', or if it has fewer tokens than the number of distinct variables in the current rule, SRV can quickly decide to disregard it.

Once this preprocessing is performed, every fragment that matches the current rule is examined and, for each such fragment, every legal literal accounted for (lines 8 through 32). Two hash tables, `positive_hash` and `negative_hash`, record the number of times each literal is associated with positive and negative examples, respectively (lines 9 through 13).

The actual search for literals is dispatched to several procedures, one for each type of literal (lines 14 through 28). Each of these procedure has the same basic functionality: All legal literals of the respective type are generated, and the count associated with each is incremented in the appropriate hash table. The specific functionality of each function is as follows:

- `SearchForLengthLits`. Recall that length literals have the form length(Relop, N). For every legal combined assignment to Relop (one of $\{<, >, =\}$) and N, the count for the corresponding length-literal is incremented in the hash table. The range of legal lengths N is determined by the range of training instance lengths and by any length-literals already added to the current rule.

- `SearchForSomeSpecializations`. Generates and accounts for all some-literals that are true for the current fragment and that use a variable introduced by a previous some-literal. Given a variable `var`, every token that can bind to it is quickly found using the table `bound`. For each such token, every combination of a path in `candidate_paths` and a simple feature in `simple_feats` corresponds to one or more some-literals, depending on whether the particular feature takes a single value or a set of values. For all distinct some-literals generated in this way for the current fragment, the corresponding count in `hash` is incremented.

- `SearchForNewSomeLits`. Like `SearchForSomeSpecializations`, only it considers introducing a new variable. Recall that SRV requires that distinct variables bind to distinct tokens. Thus, where `SearchForSomeSpecializations` iterates over tokens that can bind to a given variable, this function iterates over any token in the current fragment that is unbound in some legal binding of the variables currently used in the rule (the variable `vars`). The table `bound` facilitates the search for such tokens.

- `SearchForEveryLits`. Generates and accounts for all legal every-literals. For each feature in `simple_feats`, if all tokens in the current fragment share some value of this feature, a literal is generated.

- `SearchForPositionLits`. Given a variable `var` in use in the current rule, this procedure generates and accounts for all legal position-literals using `var`.

```
Function FirstLiteral(fieldname, docs, simple_feats, candidate_paths)
   Variable positive_hash,     /* Tables mapping to integers ... */
            negative_hash      /* ... all entries initially zero */
   For doc in docs
      For feat in simple_feats
         values = AllValuesInDoc(doc, feat)
         For value in values
            For path in candidate_paths
               tokens = TokensWithPathValue(doc, path, feat, value)
                For token in tokens
                   If TokenInField(token, fieldname, doc)
                      pcount = 1
                   Else
                      pcount = 0
                   End If
                   ncount = NonOverlapNegativeCount(token, tokens)
                   literal = "some(?A, path, feat, val)"
                   positive_hashliteral = positive_hashliteral + pcount
                   negative_hashliteral = negative_hashliteral + ncount
                End For /* tokens */
             End For    /* paths */
          End For       /* feature values */
       End For          /* simple_feats */
    End For             /* docs */
    Return FindBestLiteral(positive_hash, negative_hash)
 End Function
```

TABLE 5.4: SRV's procedure for finding the first literal in a rule.

- `SearchForRelposLits`. Given any pair of distinct variables `var1` and `var2` in use in the current rule, this procedure generates and accounts for all legal relpos-literals using `var1` and `var2`.

We cannot use variable bindings to quickly screen out candidate fragments at the beginning of rule construction, as is done in `FindNextLiteral`, because no variables have introduced. In lieu of this, however, we can speed up fragment scanning in another way: If we restrict our attention to some-literals, we can assess the effect of a particular literal without enumerating fragments. Suppose during search for the initial literal in a rule we have encountered a token that binds to '?A' in the literal:

some(?A, [ ], numericp, true)

In other words, the token consists of numerals. Suppose also that '?A' does not bind to any tokens in the proximity. Because we know the range of legal fragment sizes, we can calculate the number of fragments of which this token is a component and increment the appropriate counts for this literal, without looking at each fragment explicitly.

Because this strategy affords such a speed-up when the space to be searched is largest, and because variables introduced by some-literals are so beneficial for search efficiency, SRV uses the heuristic just described and requires that the first literal in each rule be a some-literal. The function `FirstLiteral`, depicted in Table 5.4, implements this strategy. Note that in order to avoid over-counting the number of fragments that match a literal, we must modify our calculations slightly if another numeric token occurs in close enough proximity to the one we are considering. The function `NonOverlapNegativeCount` (line 16) does this. Given a particular path-feature-value test applicable to `token`, it returns the number of fragments of which `token` is a part without counting fragments that include subsequent tokens to which this test also applies. Note also that this strategy is generally no longer applicable once a rule is non-empty, since whether a token is a component of any candidate fragments may depend on whether nearby tokens also satisfy a rule.

5.1.7 Time Complexity

In terms of time SRV is the most expensive of the learners I describe in this thesis. Its running time is $O(I \cdot M \cdot D \cdot R \cdot F)$ where:

- I is the number of instances of the target field in the training set.
- M is the size in tokens of the largest training field instance.
- D is the number of documents.
- R is the number of relational features.
- F is the number of simple features.

To arrive at this figure I make a number of assumptions:

1. The average size of a rule produced by SRV is a constant. The average rule size varies with the learning problem—rules are typically longer for harder problems—but there is no way to calculate it from SRV's inputs.

2. The size of the smallest training field instance is typically a small number that does not vary much from problem to problem. In my experiments this number was usually 1 or 2. My analysis, therefore, treats it as a constant.

3. The expense of searching for some-literals dominates that of searching for any other type of literal.

4. The expense incurred by expanding the set of candidate relational paths is more than offset by reductions in rule size and greater rule coverage. This analysis assumes that the set of candidate relational paths does not grow during rule construction, and that its size remains equal to one plus the number of relational features.

This section presents a derivation of SRV's time complexity.

Let $T_{training}$ represent SRV's training time. $T_{training}$ is linear in the number of rules SRV induces. Because no rule is allowed that covers no field instances, the number of rules is at most the number of field instances. We have, therefore:

$$T_{training} = I \cdot T_{rule}$$

where I is the number of training field instances and T_{rule} is the time SRV takes to construct a single rule.

The cost of inducing a single rule depends on the number of literals it contains. If L is the number of literals in a rule, then:

$$T_{rule} = L \cdot T_{literal}$$

In practice, there is no way to determine from the inputs how large L will be, other than to run SRV. We may speak of the average L over the set of rules learned in a training session, but this, too, is impossible to determine *a priori*. I assume, therefore, that rule size is approximately constant across learning problems (Assumption 1 above). With this assumption, L is a constant, and:

$$\begin{aligned} T_{rule} &= O(T_{literal}) \\ T_{training} &= O(I \cdot T_{literal}) \end{aligned}$$

Because finding a literal involves examining each example that matches the current rule and, for each such example, generating the set of literals applicable to it:

$$T_{literal} = O(X \cdot C)$$

where X is the number of examples and C is the number of candidate literals. The number of examples X depends on the range of legal fragment sizes. Let us make the simplifying assumption that the minimum legal size is fixed (Assumption 2 above). Because every fragment starting at a given token having up to the maximum number of tokens is an example, X is approximately the product of this maximum size M and the number of tokens in the training collection. If we posit that the number of tokens is proportional to the number of documents D in the collection we have:

$$\begin{aligned} X &\propto M \cdot D \\ T_{literal} &= O(M \cdot D \cdot C) \\ T_{training} &= O(I \cdot M \cdot D \cdot C) \end{aligned}$$

The number of candidate literals C depends on a number of factors, including the size of the set of candidate relational paths, the number of different variables used in the literals of the current rule, and the number of examples the current rule covers. These factors, however, are consequences of the current state of the search and do not depend directly on any input to SRV. In contrast, two factors do directly influence C: F, the number of simple features, and R, the number of relational features. Recall that the some-predicate has the form:

some(Var, Path, Feat, Value)

Given a particular assignment to Feat and Value, the number of candidate some-literals is the number of elements in the set of candidate relational paths, which is $(R+1)$ initially. Similarly, fix Path, and there are at least F candidate some-literals, one for each possible assignment to Feat.

With growing F and R many more some-literals will be examined than any other type of literal. The number of length-literals, position-literals, and relpos-literals is sensitive to the range in field instance lengths (i.e., to M), but it does not take many features, simple or relational, for the number of some-literals to dominate (Assumption 3). Thus, it is not too great a simplification to say:

$$\begin{aligned} C &\propto R \cdot F \\ T_{literal} &= O(M \cdot D \cdot R \cdot F) \\ T_{training} &= O(I \cdot M \cdot D \cdot R \cdot F) \end{aligned}$$

One question I do not address in this analysis is the effect of growing relational paths (Assumption 4). If SRV finds that the path [prev_token] is useful and uses it in a some-literal, it supplements the set of candidate paths with R new ones, in effect creating a potentially significant number of new candidate some-literals. Thus, whereas before any such path is used SRV considers on the order of RF literals, after one application it now has approximately $2RF$ literals to consider. In general, it must consider $O(K \cdot R \cdot F)$ literals, where K is the number of distinct relational paths used in some-literals in the current rule.

The actual impact of R on the running time of SRV depends on qualities of the particular problem that are difficult to scrutinize. If relational features *never* prove useful in learning a given field, this growth in the number of candidate literals is not a factor. In practice, even when the set of candidate paths has grown large, the costs of literal evaluation are more than offset by reduced example sets. In other words, relational features tend to pay for themselves by allowing SRV to screen out large numbers of negative examples quickly. Nevertheless, it is important to give careful thought to the number and kinds of relational features to use for a particular problem.

5.2 Experiments

I have experimented with SRV in a number of domains. Here, I present results of experiments in three of them: the seminar announcements, a collection of Web pages from university computer science departments, and a collection of newswire articles on corporate acquisitions taken from the Reuters document collection (Lewis, 1992).

For the purposes of comparison with other algorithms described in the thesis, the seminar announcement experiments are the most relevant. The other two domains, however, serve as illustrations of SRV's extensibility. Adapting SRV for use in these domains only

word	single_digit_p	long_char_p	punctuationp
singletonp	double_char_p	long_digit_p	hybrid_anum_p
doubletonp	double_digit_p	capitalized_p	a_then_num_p
tripletonp	triple_char_p	all_upper_case_p	num_then_a_p
quadrupletonp	triple_digit_p	all_lower_case_p	multi_word_cap_p
longp	quadruple_char_p	numericp	*prev_token*
single_char_p	quadruple_digit_p	sentence_punct_p	*next_token*

TABLE 5.5: SRV's default features.

involves adding several domain-specific features to its default feature set: features that exploit HTML, in the case of the Web pages; features that reflect syntactic and lexical information, in the case of the acquisition articles. Each such feature is implemented as a C function, which is compiled into SRV.

The comparison of SRV with other algorithms favors SRV. SRV's performance is best on most of the fields. On the few fields on which SRV does not perform best, its performance is close to best.

5.2.1 Case Study: Seminar Announcements

Experiments with SRV in the seminar announcements domain followed the same regime as in previous chapters. The same five partitions of the document collection as in Chapters 3 and 4—half for training, half for testing—were used to train and test SRV. Performance numbers are averages over the five iterations.

Table 5.5 shows the token features that were made available to SRV in these experiments. The word feature returns the literal token, modulo capitalization—unless the token occurs fewer than five times in the training set, in which case it returns the value *unknown*. Here, longp means longer than four characters in length. Except for *prev_token* and *next_token*, which are relational, each of these features is a simple feature. The set of simple features is exactly the same as that used in alphabet transduction in Chapter 4. Because all of the features in the figure are applicable in any domain, I call this SRV's "default" feature set.

Figures 5.3 through 5.6 show rules learned by SRV for the *speaker*, *location*, *stime*, and *etime* fields, respectively. Each rule is the one from the first train/test partition achieving the highest validation score; in fact, all four rules had no false matches. The relative difficulty of each field is reflected in part by the number of validation examples the corresponding rule matches: The *speaker* rule matched 9, the *location* rule 15, the *stime* rule 46, and the *etime* rule 33 validation examples. The equivalent of each rule in first-order logic is also shown in each table. At the bottom of each Table is a matching fragment. Below each such fragment are positioned the variables used in the rule, each one below the token to which it binds. Recall that the word feature returns *unknown* if a token occurs fewer than five

```
speaker :-
    some(?A, [ ], word, *unknown*),
    every(capitalizedp, true),
    length(=, 2),
    some(?B, [ ], word, *unknown*),
    some(?B, [prev_token], word, ":"),
    some(?A, [next_token], doubletonp, false),
    every(quadruple_char_p, false),
    some(?B, [prev_token prev_token], word, "who")
```

$$
\begin{array}{rcl}
\textit{speaker}(F) & \leftarrow & \textit{length}(F) = 2 \\
 & \wedge & (\forall A \in F.\textit{capitalizedp}(F) = \textit{true} \\
 & & \wedge\ \textit{quadruple\_char\_p}(A) = \textit{true}) \\
 & \wedge & (\exists A, B \in F.A \neq B \\
 & & \wedge\ \textit{word}(A) = \texttt{*unknown*} \\
 & & \wedge\ \textit{doubletonp}(\textit{next\_token}(A)) = \textit{false} \\
 & & \wedge\ \textit{word}(B) = \texttt{*unknown*} \\
 & & \wedge\ \textit{word}(\textit{prev\_token}(B)) = \texttt{`:'} \\
 & & \wedge\ \textit{word}(\textit{prev\_token}(\textit{prev\_token}(B))) = \texttt{`who'})
\end{array}
$$

Fragment F is a *speaker* if:
F contains a novel token (A); and
every token in F is capitalized; and
F contains exactly two tokens; and
F contains another novel token (B); and
B is preceded by a colon; and
A is not followed by a two-character token; and
every token in F does *not* consist of
exactly four alphanumeric characters; and
two tokens before B is the word "who"

```
Who:  [Toshinari Takahashi] &lt;takahasi@...
         ?B         ?A
```

FIGURE 5.3: A rule learned by SRV to recognize instances of the *speaker* field, its first-order logic equivalent, its English translation, and a fragment of text it matches.

```
location :-
    some(?A, [ ], word "hall"),
    length(=, 3),
    every(quadrupletonp, true),
    some(?B, [ ], quadruple_digit_p, true),
    position(?A, fromlast, =, 1)
```

$$
\begin{array}{rcl}
\textit{location}(F) & \leftarrow & \textit{length}(F) = 3 \\
 & \wedge & (\forall A \in F.\textit{quadrupletonp}(A) = \textit{true}) \\
 & \wedge & (\exists A, B \in F.A \neq B \\
 & & \wedge\ \textit{word}(A) = \texttt{`hall'} \\
 & & \wedge\ \textit{quadruple\_digit\_p}(B) = \textit{true} \\
 & & \wedge\ \textit{dist\_from\_last}(A, F) = 1)
\end{array}
$$

Fragment F is a *location* if:
F contains the word "hall" (A); and
F contains exactly three tokens; and
every token in F consists of exactly four characters; and
F contains a four-digit token (B); and
A is the next-to-last token

```
Location:  [Wean Hall 4601]
                 ?A   ?B
```

FIGURE 5.4: A rule learned by SRV to recognize instances of *location*, its equivalent in first-order logic, its English translation, and a matching fragment with variable bindings.

```
stime :-
   some(?A, [prev_token], word, “:”),
   position(?A, fromfirst, =, 2),
   some(?A, [prev_token prev_token], numericp, true),
   length(<, 5),
   some(?B, [prev_token prev_token], quadrupletonp, true),
   some(?C, [ ], word, “pm”),
   some(?B, [prev_token], singletonp, true)
```

$$
\begin{array}{lcl}
\mathit{stime}(F) & \leftarrow & \mathit{length}(F) < 5 \\
 & \wedge & (\exists A, B, C \in F . A \neq B \neq C \\
 & & \wedge\ \mathit{word}(\mathit{prev\_token}(A)) = \texttt{':'} \\
 & & \wedge\ \mathit{numericp}(\mathit{prev\_token}(\mathit{prev\_token}(A))) = \mathit{true} \\
 & & \wedge\ \mathit{quadrupletonp}(\mathit{prev\_token}(\mathit{prev\_token}(B))) = \mathit{true} \\
 & & \wedge\ \mathit{singletonp}(\mathit{prev\_token}(B)) = \mathit{true} \\
 & & \wedge\ \mathit{word}(C) = \texttt{'pm'})
\end{array}
$$

Fragment F is a *stime* if:
- F contains a token (A) preceded by a colon; and
- A is the third token in F; and
- two tokens before A is a numeric token; and
- F contains fewer than five tokens; and
- F contains a token (B), two tokens before which is a four-character token; and
- F contains the word “pm” (C); and
- B is preceded by a single-character token

```
Time:  [5:00 PM]
        ?B ?A ?C
```

FIGURE 5.5: A rule learned by SRV to recognize instances of *stime*, its equivalent in first-order logic, its English translation, and a matching fragment.

```
etime :-
   some(?A, [ ], double_digit_p, true),
   position(?A, fromfirst, =, 2),
   some(?A, [prev_token], word, ”:”),
   some(?B, [prev_token prev_token], word, ”-”),
   length(<, 5),
   some(?C, [ ], double_char_p, true),
   some(?C, [ ], all_lower_case, false)
```

$$
\begin{array}{lcl}
\mathit{etime}(F) & \leftarrow & \mathit{length}(F) < 5 \\
 & \wedge & (\exists A, B, C \in F . A \neq B \neq C \\
 & & \wedge\ \mathit{double\_digit\_p}(A) = \mathit{true} \\
 & & \wedge\ \mathit{word}(\mathit{prev\_token})(A) = \texttt{':'} \\
 & & \wedge\ \mathit{dist\_from\_first}(A) = 2 \\
 & & \wedge\ \mathit{word}(\mathit{prev\_token}(\mathit{prev\_token}(B))) = \texttt{'-'} \\
 & & \wedge\ \mathit{double\_char\_p}(C) = \mathit{true} \\
 & & \wedge\ \mathit{all\_lower\_case}(C) = \mathit{false})
\end{array}
$$

Fragment F is an *etime* if:
- F contains a two-digit token (A); and
- A is the third token in F; and
- A is preceded by a colon; and
- F contains a token (B), two tokens before which is a hyphen; and
- F contains fewer than five tokens; and
- F contains a two-letter token (C); and
- C does not consist entirely of lower-case letters

```
Time:  3:30 - [5:00 PM]
              ?B?A  ?C
```

FIGURE 5.6: A rule learned by SRV to recognize instances of *etime*, its equivalent in first-order logic, and a matching fragment.

	speaker			location			stime			etime		
	F1	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*
Rote	12.1	55.1	6.8	70.6	90.1	58.1	73.9	74.8	73.0	53.6	53.1	54.1
BayesIDF	29.7	41.8	23.0	61.3	66.3	57.0	98.2	98.2	98.2	92.3	94.6	90.1
BayesGI	52.4	60.6	46.1	71.4	79.1	65.0	96.3	98.5	94.1	89.9	96.5	84.0
SRV	**58.4**	60.9	56.1	73.3	82.2	66.1	98.4	98.4	98.3	**94.1**	98.4	90.1

TABLE 5.6: Peak F1 scores, with corresponding precision and recall, for all methods on all seminar announcement fields.

	speaker		location		start time		end time	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
Rote	55.1	6.8	89.5	58.1	73.7	73.4	37.4	71.6
BayesIDF	28.8	27.4	57.3	58.8	98.2	98.2	46.8	95.7
BayesGI	46.0	53.3	68.1	66.6	94.4	94.2	39.7	84.8
SRV	54.9	58.3	73.8	69.5	98.4	98.3	66.7	92.6

TABLE 5.7: Precision and recall of all methods on all seminar announcement fields.

times in the training set

Tables 5.6 (peak F1 scores) and 5.7 (precision and recall) compare all four learners developed so far in terms of performance on the four seminar announcement fields. Recall that F1 scores in bold face show the best learner with 95% confidence. Figures 5.10 through 5.13 (at the end of the chapter) depict the same performance results with precision/recall graphs. In all cases, SRV is among the best methods, and in the case of *speaker* and *location* it is to be preferred. Table 5.6 makes this clear; SRV achieves the best peak F1 score on all fields. (Recall that the *peak* score is calculated by choosing the point on the precision/recall curve for which the F1 score is maximized.) Note that BayesGI is competitive with SRV on all fields. This is in spite of the poor performance of the canonical acceptor in isolation, and the relative weakness of BayesIDF on the speaker and location fields.

5.2.2 Case Study: Web Pages

Because it makes no special assumptions about document characteristics, SRV can be used "out of the box" for extraction from Web pages. Given only its default feature set, it will learn effective rules. In most cases, however, in order to maximize its performance it needs access to information reflecting the HTML-specific structure in a page. In this section, I describe an experiment in the definition of features to reflect this information.

in_title	in_list	in_center	after_p	*table_next_col*
in_a	in_tt	in_strong	after_li	*table_prev_col*
in_h	in_table	in_em	after_td	*table_next_row*
in_h1	in_b	in_emphatic	after_th	*table_prev_row*
in_h2	in_i	after_br	after_td_or_th	*table_row_header*
in_h3	in_font	after_hr		*table_col_header*

TABLE 5.8: HTML features added to SRV's default feature set. Features in italics are relational.

Feature Definition

The purpose of most HTML tags is to define a scope over parts of the text visible to the user. One way of rendering such information in a form usable by SRV is to associate a Boolean feature with each kind of tag. For tokens occurring within the scope of a tag, the corresponding feature returns *true*; for tokens occurring outside the scope, it returns *false*. For example, to the `<title>` tag we can assign a feature, "in_title," the value of which is *false* except for words occurring in a page's title field. Note that this approach does not treat tags themselves as tokens, but as features of the text that is actually visible. (This is in part a consequence of the document representation shared by all learners. Refer to Appendix C for a discussion of this representation.)

In addition to tags that define scopes, HTML allows for a number of tags that can be used to delimit without enclosing, such as `<li>` and `<p>`. I defined a Boolean feature for each of these. In contrast with features defined for enclosing tags, these features return *true* only for the single token immediately following the corresponding tag. The after_p feature, for example, returns *true* only for tokens at the beginning of text blocks initiated with `<p>` tags.

Tables are a frequently used presentation device in HTML. In order to understand a table entry, it may be necessary to examine entries in adjacent rows or columns, or to read the row or column header. I defined a number of relational features which allow SRV to navigate tables in this way. These features, along with all the other features defined for these experiments, are shown in Table 5.8. Each such feature maps to the first token in the corresponding table cell. For example, the feature table_row_header returns the first token in the row header cell.

Domains

The document collection used in the Web page experiments was assembled as part of the World Wide Knowledge Base (WebKB) project (Craven *et al.*, 1998). As part of preliminary experiments, WebKB sampled a large number of pages from four university computer science departments: Cornell University, University of Texas, University of Washington, and University of Wisconsin. Each such page was manually assigned to one of several disjoint classes.

I sampled pages from two of the WebKB classes—101 pages from the course class, 96 from the research project class—and annotated instances of five fields:

- *crsTitle*, the title of a course in a course page
- *crsNumber*, the official number of a course in a course page
- *crsInst*, the names of any instructors and teaching assistants in a course page
- *projTitle*, the project title in a project page
- *projMember*, the names of any members and alumni of a project in a project page

Additional details about this domain are available in Appendix A.

Methodology and Evaluation

Two separate experiments were conducted, one each for the course and project page collections. In each experiment I followed the same regime as in experiments with the seminar announcements: five random partitions of a collection into training and testing sets of equal size.

Two of the fields in this corpus, *course instructor* and *project member*, violate the "one-field-per-document" (OPD) assumption that underlies the performance metrics used so far in this thesis. Each of these two fields usually has multiple distinct instantiations in a single document. The *course instructor* field was defined to include teaching assistants; thus, in a typical course page at a large university several people qualify as course instructors. Similarly, a research project usually has multiple members, all of which are listed on its official page. I call such fields "many-per-document" (MPD) fields.

Given a MPD field the central question cannot be whether a learner's top prediction identified a field instance. Instead, we want to know what fraction of the field instances it identified, and what fraction of its predictions identified a field instance. Therefore, for MPD fields, *Recall* is the fraction of all field instances covered by some prediction. *Precision* is the fraction of all predictions that covered some field instance.

Results

Table 5.9 shows the peak F1 scores achieved by all learners, including SRV with HTML features, on the OPD fields. Table 5.10 shows the precision and recall (at full recall) on the same fields. And Figures 5.14 through 5.16 (located at the end of the chapter) present the full precision/recall curves. It is immediately apparent from these results that SRV achieves stronger performance than the other learners in general. The only field for which this is not obviously true is *crsNumber*, for which the performance of BayesGI is slightly better.

Peak F1 scores for the MPD fields are shown in Table 5.11, precision and recall at full recall are shown in Table 5.12, and the full precision/recall curves are presented in

	crsNumber			crsTitle			projTitle		
	F1	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*
Rote	33.1	70.8	21.6	32.3	52.1	23.4	15.0	39.1	9.3
BayesIDF	58.2	60.2	56.4	47.7	56.8	41.1	11.7	13.6	10.3
BayesGI	**89.9**	89.5	90.3	39.6	57.0	30.4	17.1	17.8	16.5
SRV (default)	86.7	86.2	87.3	35.6	38.4	33.2	8.8	8.4	9.3
SRV (HTML)	87.2	86.3	88.1	**55.9**	83.3	42.1	**33.7**	41.7	28.4

TABLE 5.9: Peak F1 scores of three learners on the three "one-per-document" fields from the Web domain.

	crsNumber		crsTitle		projTitle	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
Rote	70.8	21.6	44.3	23.8	37.5	9.3
BayesIDF	56.4	59.7	38.0	44.4	10.6	12.9
BayesGI	85.2	90.3	31.2	36.4	14.0	17.0
SRV (default)	84.1	87.3	33.0	34.6	7.7	9.3
SRV (HTML)	84.0	89.0	45.7	50.0	26.3	31.4

TABLE 5.10: Precision and recall of all learners, including SRV with HTML features, on the three OPD fields of the WebKB domain.

	crsInst			projMember		
	F1	*Prec*	*Rec*	*F1*	*Prec*	*Rec*
Rote	20.4	71.8	11.9	21.4	76.4	12.4
BayesIDF	32.8	30.8	35.1	18.8	19.6	18.0
BayesGI	**48.1**	50.3	46.0	35.4	41.3	31.0
SRV (default)	37.6	36.2	39.2	34.8	32.8	37.0
SRV (HTML)	42.9	49.1	38.0	**41.4**	44.2	39.0

TABLE 5.11: Peak F1 scores of all learners, including SRV with HTML features, on the two "many-per-document" fields from the WebKB domain.

	crsInst		projMember	
	Prec	*Rec*	*Prec*	*Rec*
Rote	71.8	11.9	74.8	11.4
BayesIDF	8.7	48.3	15.4	19.1
BayesGI	7.7	59.6	23.7	38.4
SRV (default)	20.6	53.2	26.2	35.2
SRV (HTML)	21.6	55.9	30.0	41.0

TABLE 5.12: Precision and recall of all learners, including SRV with HTML features, on the “many-per-document” fields of the WebKB domain.

Figures 5.17 and 5.18 at the end of the chapter. While SRV performs best on only one of these fields (*projMember*; BayesGI performs better on *crsInst*), its results on *crsInst* are nevertheless competitive with those of BayesGI. Note, too, that BayesGI suffers a sharp drop in precision at the high-recall end of its curve for *crsInst* in Figure 5.17.

A comparison between SRV with and without HTML features strongly favors the version with the features. In this domain there is clear evidence that SRV can exploit genre-specific information.

5.2.3 Case Study: Newswire Articles

To this point I have only presented results for domains in which linguistic analysis is difficult or impossible. However, there is no reason why Rote, BayesIDF, and SRV cannot be applied to linguistically well-structured domains. While Rote and BayesIDF, as implemented, will ignore any linguistic structure, SRV is poised to exploit it. In this section, I relate an experiment in which these three learners are applied to a domain consisting of newswire articles on corporate acquisitions. A central question addressed in these experiments is how well SRV can make use of the linguistic information provided by two off-the-shelf NLP packages, one a syntactic parser, the other a semantic lexicon.

Linguistic Syntax: the Link Grammar Parser

Syntactic information is provided by the link grammar parser (Sleator and Temperley, 1993). This parser takes a sentence as input and returns a complete parse in which terms are connected in typed binary relations (“links”) which represent syntactic relationships. I mapped each such link to a relational feature: A token on the right side of a link of type X has a corresponding relational feature called left_X that maps to the token on the left side of the link. In addition, several non-relational features, such as part of speech, are derived from parser output.

Figure 5.7 shows part of a link grammar parse and its translation into features. The tokens “Corp” and “said” share an S-link, which is used to connect the subject of a clause

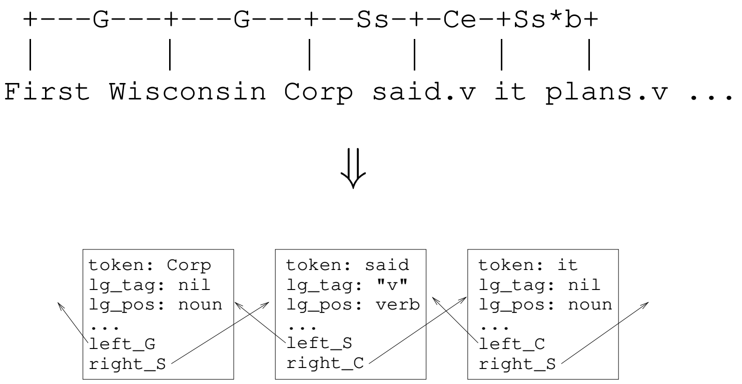

FIGURE 5.7: An example of link grammar feature derivation.

with the main verb. There are two relational features that correspond to the S-line, right_S and left_S. The value of right_S for the "Corp" token is the "said" token; similarly, a left_S feature points from the "said" token to the "Corp" token.[5] Based on the output of the parser shown in the figure, we can infer that the "Corp" token is a noun; lg_pos, a simple feature representing part of speech, takes the value noun for this token. The parts of speech of other tokens are inferred in a similar way.

Lexical Semantics: Wordnet

My object in using Wordnet (Miller, 1995) is to enable SRV to recognize that the phrases, "A bought B," and, "X acquired Y," are instantiations of the same underlying pattern. Although "bought" and "acquired" do not belong to the same "synset" in Wordnet, they are nevertheless closely related in Wordnet by means of the "hypernym" (or "is-a") relation. To exploit such semantic relationships I created a single token feature, called wn_word. In contrast with features already outlined, which are mostly boolean, this feature is set-valued. For nouns and verbs, its value is the set of all synsets in the hypernym path to the root of the hypernym tree in which a word occurs. For adjectives and adverbs it is a cluster of closely related synsets. In the case of multiple Wordnet senses, I used the most common sense of a word, according to Wordnet, to construct this set.

Figure 5.8 displays the hypernym path from one sense of the noun "acquisition" to the root of the hierarchy to which it belongs. The value of the wn_word feature for "acquisition" would therefore be:

wn_word("acquisition") = { acquisition, deed, accomplishment, action, act }

[5]Note that a feature such as left_S can be queried even for tokens not connected by S-links. The value of such a feature is *undefined*. Thus, by predicating on this *undefined* value, SRV can exploit the *absence* of specific links. For example, in Figure 5.7 the value of left_S for the token "Wisconsin" is *undefined*, because this token is not attached to any S-link.

```
acquisition, acquiring, getting -- (the act of contracting
       => deed, feat, effort, exploit -- (a notable achieve
           => accomplishment, achievement -- (the act of ac
               => action -- (something done (usually as opp
                   => act, human action, human activity --
```

FIGURE 5.8: One sense of the word "acquisition" and all its generalizations in Wordnet.

where each word in the set stands for the Wordnet synset to which it belongs. Suppose that SRV needed to merge the words "acquisition" and "completion." The wn_word feature might be used thus:

some(?A, [ ], wn_word, "accomplishment")

Because the synset to which "accomplishment" belongs is a generalization of both "acquisition" and "completion," this literal would hold for fragments containing either word.

Domain

To construct the domain for these experiments, I sampled 600 articles belonging to the "acquisition" category in the Reuters corpus (Lewis, 1992) and tagged them for ten fields, nine of which are used in these experiments. Fields include those for the official names of the parties to an acquisition (acquired, purchaser, seller), as well as their short names (acqabr, purchabr, sellerabr), the location of the purchased company or resource (acqloc), the price paid (dlramt), and any short phrases summarizing the progress of negotiations (status). The fields vary widely in length and frequency of occurrence, both of which have a significant impact on the difficulty they present for learners. For more details on this domain, please refer to Appendix A.

Results

Table 5.13 gives the peak F1 scores of Rote, BayesIDF, and SRV, both with and without linguistic features on nine of the acquisitions fields. Table 5.14 presents the precision and recall of the same learners. Figures 5.19 through 5.27 (end of chapter) show the full precision/recall curves. It is evident from these results that the acquisitions domain poses a considerably stiffer challenge for all the learning methods than either the seminar announcements or the Web pages. Lacking are the clear signals provided by simple presentational devices, as in the seminar announcements, or HTML structure, as in the Web pages. The acquisition articles do contain highly conventional language in many cases. In order to perform well, however, it appears that some representation of the semantic content of the articles is required. For example, SRV does relatively well at identifying companies, but often the patterns it exploits are too weak to disambiguate instances of *acquired* from

	F1	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*
	acquired			*purchaser*			*seller*		
Rote	18.9	66.5	11.0	17.4	43.9	10.8	17.2	41.7	10.8
BayesIDF	20.2	21.7	19.0	39.5	40.0	39.0	**28.5**	28.9	28.0
SRV	38.5	42.7	35.0	45.1	47.4	43.0	23.4	21.3	26.1
SRVlng	37.8	39.8	36.0	44.2	48.0	41.0	23.1	19.6	28.0
	acqabr			*purchabr*			*sellerabr*		
Rote	17.0	37.5	11.0	13.5	9.0	26.4	9.8	15.7	7.1
BayesIDF	29.8	34.8	26.0	47.4	47.8	47.0	**31.8**	29.0	35.1
SRV	38.1	37.0	39.2	48.5	44.7	53.0	25.1	20.7	32.1
SRVlng	**42.7**	40.5	45.0	49.9	45.6	55.0	22.6	17.0	34.0
	acqloc			*status*			*dlramt*		
Rote	10.3	10.6	10.0	**49.6**	50.3	49.0	48.7	67.4	38.1
BayesIDF	20.7	24.1	18.1	41.3	43.9	39.0	52.6	58.2	48.0
SRV	22.3	22.7	22.0	47.0	59.1	39.0	61.8	66.1	58.1
SRVlng	21.7	21.3	22.0	47.2	57.4	40.0	60.6	56.8	65.0

TABLE 5.13: Peak F1 scores, and corresponding precision and recall, of Rote, BayesIDF, SRV, and SRV augmented with linguistic features on nine of the acquisitions fields.

	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
Alg	*acquired*		*purchaser*		*seller*	
Rote	59.6	11.2	43.9	10.8	41.7	10.8
BayesIDF	19.8	20.0	36.9	40.4	15.6	38.7
SRV	38.4	37.5	42.9	45.9	16.3	33.6
SRVlng	38.0	36.6	42.4	44.6	16.4	32.8
	acqabr		*purchabr*		*sellerabr*	
Rote	22.1	12.0	16.8	9.4	9.8	7.8
BayesIDF	23.2	32.1	39.6	52.9	16.0	51.5
SRV	31.8	43.8	41.4	55.0	14.2	42.8
SRVlng	35.5	48.3	43.2	57.0	14.7	42.6
	acqloc		*status*		*dlramt*	
Rote	6.4	12.4	42.0	50.7	63.2	38.8
BayesIDF	7.0	23.6	33.3	43.6	24.1	54.5
SRV	12.7	29.4	39.1	44.7	50.5	69.8
SRVlng	15.4	28.8	41.5	46.5	52.1	68.5

TABLE 5.14: Precision and recall at full recall of Rote, BayesIDF, SRV, and SRV augmented with linguistic features on nine of the acquisitions fields.

instances of *purchaser*. To perform such disambiguation it might be necessary to analyze the semantics of a sentence in order to sort out the respective roles of the companies it contains.

It is surprising that BayesIDF does as well as it does. On several fields the results argue for a preference of BayesIDF over either version of SRV. Note, however, that these experiments used an early version of SRV which included no every-predicate. Subsequent experiments on this domain (to be presented in the next chapter) show that availability of the every-predicate lifts SRV's performance above that of BayesIDF.[6]

SRV is able to use the linguistic features to good effect in some cases—particularly on the *acqabr* field. Figure 5.9 presents a rule for learned for *acqabr* in which both link grammar and Wordnet features are used. The AN-link connects a noun modifier with the noun it modifies. In the fragments in the figure, both "Trilogy" and "Roach" are connected to "shares" by such a link. In Wordnet "possession" is a hypernym of "stock." The fact that the relational paths [right_AN] and [right_AN prev_token] both point to words belonging to the same hypernym path in Wordnet indicates the presence of a Wordnet collocation. The path to the top of the hypernym tree is shown below the collocation "treasury shares" in the figure. Note how two of the synsets on this path are used in the rule, first the more general "possession," then the more specific "stock." In effect, the linguistic literals in this rule stipulate that an *acqabr* fragment should be a noun used to modify a two-word collocation meaning "stock."

Overall, however, the introduction of linguistic information appears to have little effect on SRV's performance. We might expect the relational features derived from the link grammar to help by giving SRV access to *non-local* syntactically meaningful patterns. Instead, it seems that patterns found by following the default relational features are just as powerful. Similarly, we look to Wordnet to help SRV generalize over synonyms, but the results suggest that the language of the articles is stylized enough that synonymy is not a problem for SRV without lexical information.

I conducted a three-fold experiment to investigate the effect of each kind of linguistic information, syntactic and lexical. In one case, SRV was trained with all the link-grammar features but without wn_word. In another, it was given wn_word but no link-grammar features. As Figure 5.15 shows, results are mixed. For some fields, such as *acqabr*, both syntactic and lexical information appear to provide some benefit, combining to yield even better performance than either alone. For others, such as *dlramt*, either component alone results in worse precision than the combination at the same or worse recall For still others, such as *purchaser*, it seems that any kind of linguistic information provides at best marginal benefit.

It would seem then that SRV cannot consistently turn additional features into stronger performance. Although it is not entirely clear why this is so, two candidate reasons suggest themselves. First, it may be that SRV's strong local bias, its search from small fragments

[6]Because the experiments involving linguistic features were inordinately expensive in terms of time (in part due to many syntactic relational features) and yielded mediocre results, I have not re-run them with the every-predicate.

```
acqabr :-
  some(?A, [ ], capitalized, true)
  length(<, 2)
  some(?A, [next_token], all_lower_case, true)
  some(?A, [prev_token], all_lower_case, true)
  some(?A, [right_AN], wn_word, “possession”)
  some(?A, [right_AN prev_token], wn_word, “stock”)
  some(?A, [prev_token prev_token], doubleton, false)
```

$$
\begin{array}{lll}
\mathit{acqabr}(F) & \leftarrow & \mathit{length}(F) < 2 \\
 & \wedge & (\exists A \in F.\mathit{capitalized}(A) = \mathit{true} \\
 & & \wedge\ \mathit{all\_lower\_case}(\mathit{next\_token}(A)) = \mathit{true} \\
 & & \wedge\ \mathit{all\_lower\_case}(\mathit{prev\_token}(A)) = \mathit{true} \\
 & & \wedge\ \texttt{'possession'} \in \mathit{wn\_word}(\mathit{right\_AN}(A)) \\
 & & \wedge\ \texttt{'stock'} \in \mathit{wn\_word}(\mathit{prev\_token}(\mathit{right\_AN}(A))) \\
 & & \wedge\ \mathit{doubleton}(\mathit{prev\_token}(\mathit{prev\_token}(A))) = \mathit{false})
\end{array}
$$

Fragment F is an *acqabr* if:
- F contains a capitalized token (A); and
- F contains fewer than two tokens; and
- A is followed by a lower-case token; and
- A is preceded by a lower-case token; and
- A is connected by an AN-link to a token with the Wordnet meaning “possession”; and
- A is connected to a token, which is preceded by a token with the Wordnet meaning “stock”; and
- two tokens before A is a token that does not consist of exactly two characters

```
                        +------AN------+
                        |              |
to purchaser 4.5 mln [Trilogy] common shares at
acquired another 2.4 mln [Roach] treasure shares
                                      ∩
                                    stock
                                      ∩
                                   capital
                                      ∩
                                    asset
                                      ∩
                                  possession
```

FIGURE 5.9: A learned rule for *acqabr* that uses linguistic features, along with two fragments of matching text and relevant linguistic information.

	purchaser		acqabr		dlramt		status	
	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
Rote	44.2	10.6	22.4	11.1	59.3	33.7	38.5	47.5
BayesIDF	35.8	39.3	24.3	32.5	22.6	53.8	31.8	41.9
SRV (def)	41.8	44.8	33.9	45.3	49.2	71.4	38.4	45.2
SRV (ling)	42.2	44.5	37.6	49.4	52.3	67.6	39.7	46.2
SRV (lg)	42.4	45.2	35.3	46.9	41.6	67.6	34.8	41.1
SRV (wn)	41.4	44.4	34.9	45.7	47.3	64.2	38.7	44.3

TABLE 5.15: Precision and recall results from a three-fold experiment on four fields for the three basic learners, plus SRV with syntactic and lexical information (SRV (ling)), SRV with only syntactic information (SRV (lg)), and SRV with only lexical information (SRV (wn)).

outwards, renders linguistic information less useful than it otherwise might be. Soderland, in the first investigation into learning rules for information extraction, took complete, parsed sentences as his example representation (Soderland, 1996). Of course, this representation is difficult or impossible for some of the domains I investigate. Where appropriate, however, it constitutes a powerful bias. At the least, it provides more context in which to look for linguistic patterns. The second possible reason for SRV's inconsistency is noise. Both of the NLP packages have limited coverage and are not tuned to the acquisitions corpus. Consequently, it is almost inevitable that some of the information they produce is incorrect. My hypothesis that learning would nevertheless find useful signal in their output was not born out by the experiments.

5.3 Discussion

By all measures, SRV is the best of the learning algorithms for information extraction described in this thesis. While it does not perform the best on all fields—on a few it is second best—it is the most consistently strong performer. The relational learning paradigm gives it great flexibility in the kinds of patterns it can express. In contrast with BayesGI, which uses token features in a processing step separate from the primary induction step, SRV's use of such features is a direct part of its search for good extraction patterns.

And SRV is able to make good use of features that reflect genre-specific information. When its default, domain-independent feature set is augmented with HTML-specific features, SRV shows substantial performance gains. This is perhaps the most attractive aspect of SRV, aside from its comparatively strong performance. It is useful in a wide range of domains, and its usefulness in any particular domain can be increased with the aid of a few suitably designed features.

Note, however, that although SRV almost always performed better than the other learners, it did not solve any new fields outright. Those fields for which its performance is es-

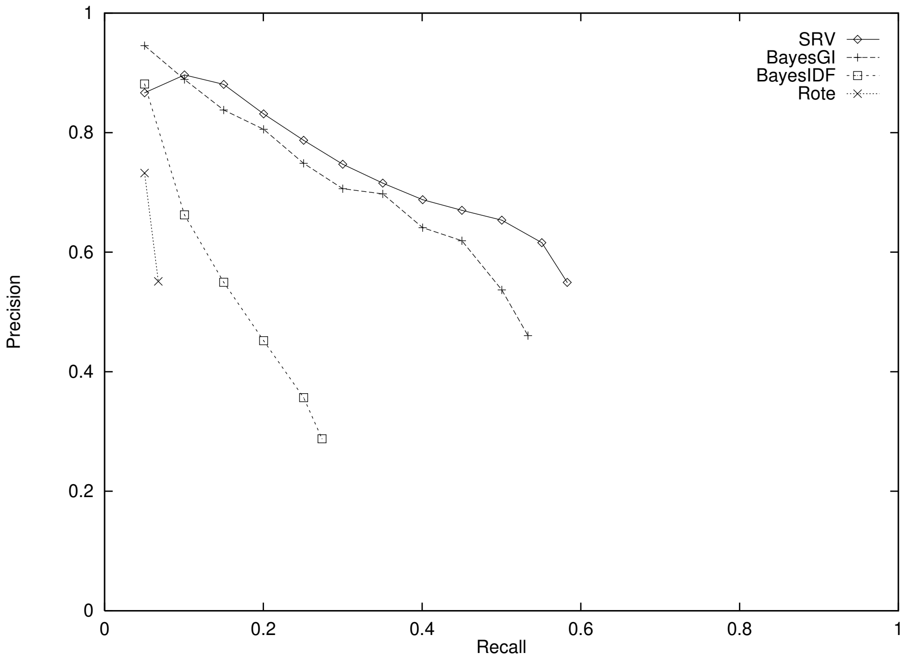

FIGURE 5.10: Precision/recall plot comparing all four learners on the *speaker* field from the seminar announcement domain.

sentially perfect are also those fields effectively solved by simpler learners. In most cases there is plenty of residual error left if SRV's performance, even if there is less of it than for other learners. The next chapter shows how to take steps toward reducing this error.

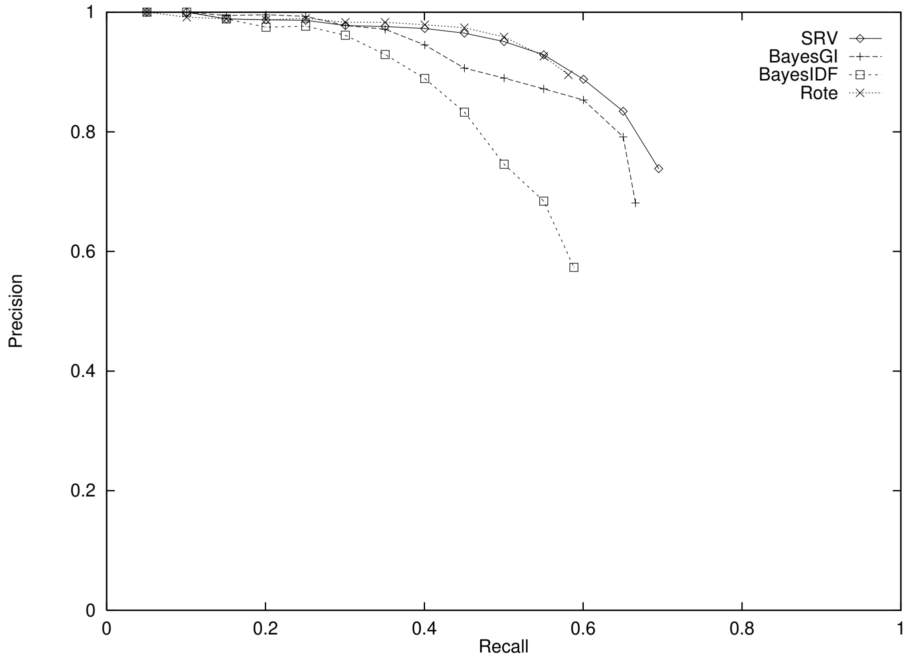

FIGURE 5.11: Precision/recall plot comparing all four learners on the *location* field from the seminar announcement domain.

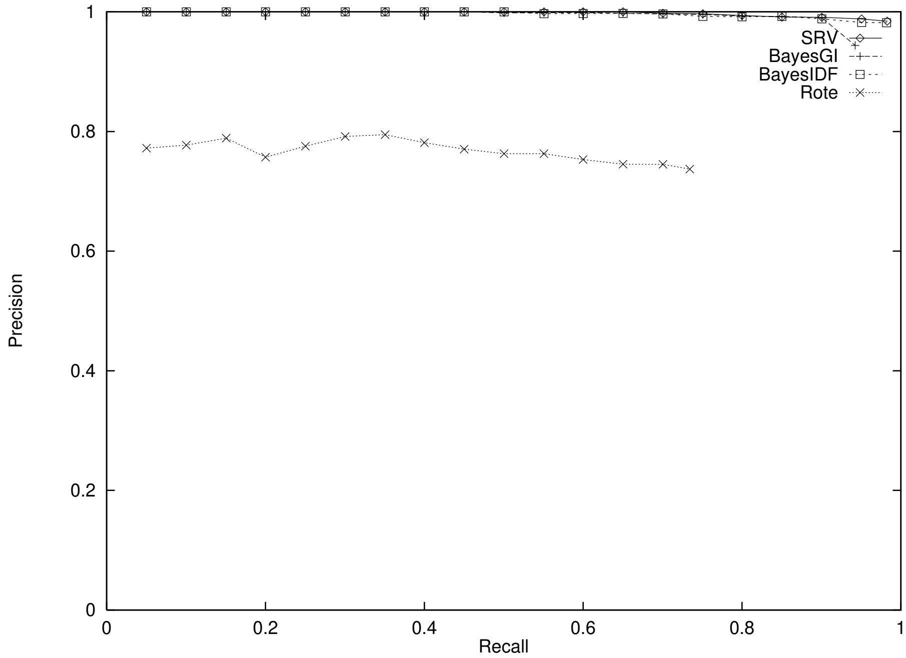

FIGURE 5.12: Precision/recall plot comparing all four learners on the *stime* field from the seminar announcement domain.

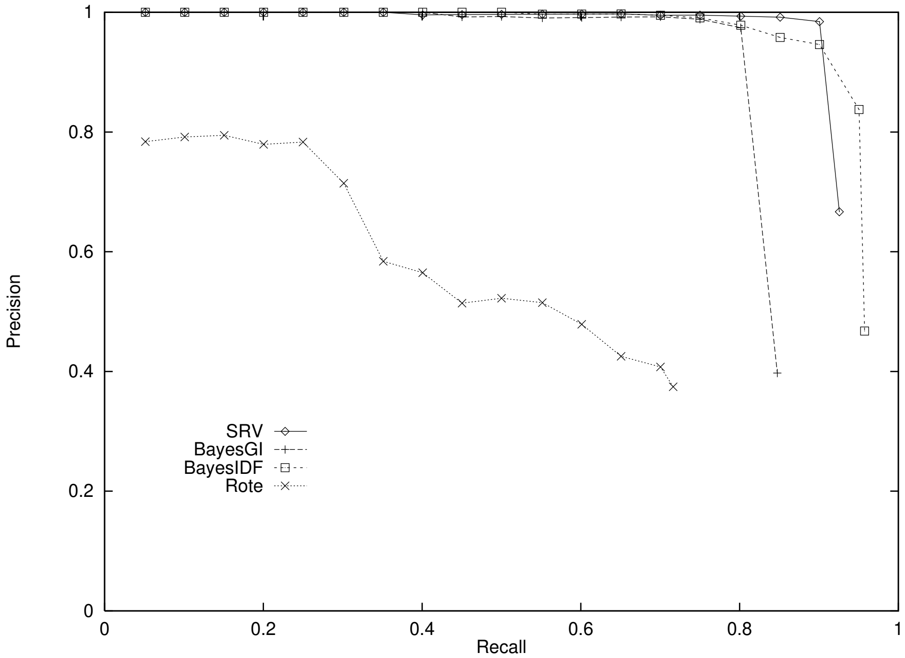

FIGURE 5.13: Precision/recall plot comparing all four learners on the *etime* field from the seminar announcement domain.

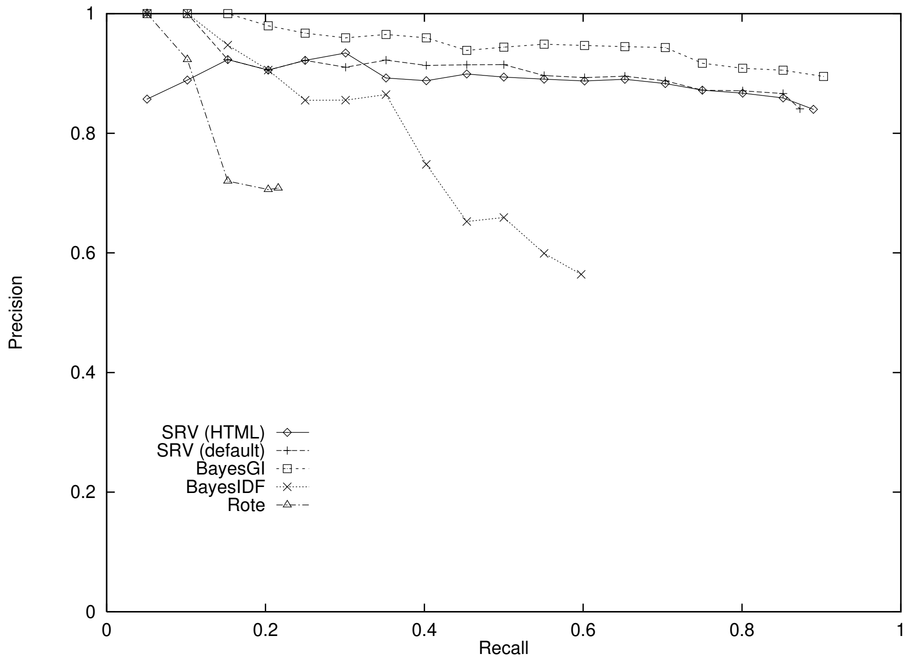

FIGURE 5.14: Precision/recall plot comparing all four learners (include SRV without HTML features) on the *crsNumber* field from the WebKB domain.

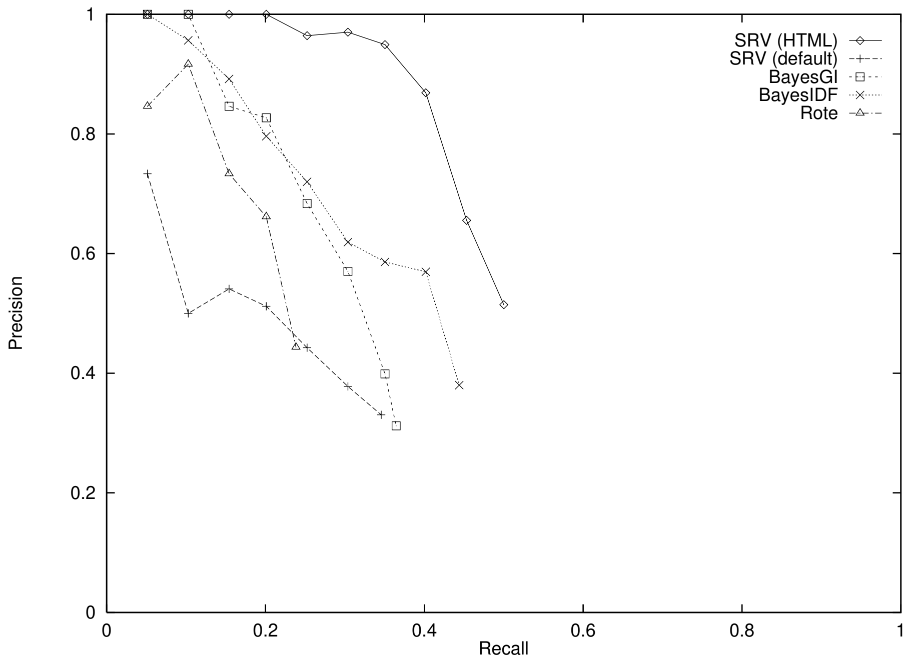

FIGURE 5.15: Precision/recall plot comparing all four learners (including SRV without HTML features) on the *crsTitle* field from the WebKB domain.

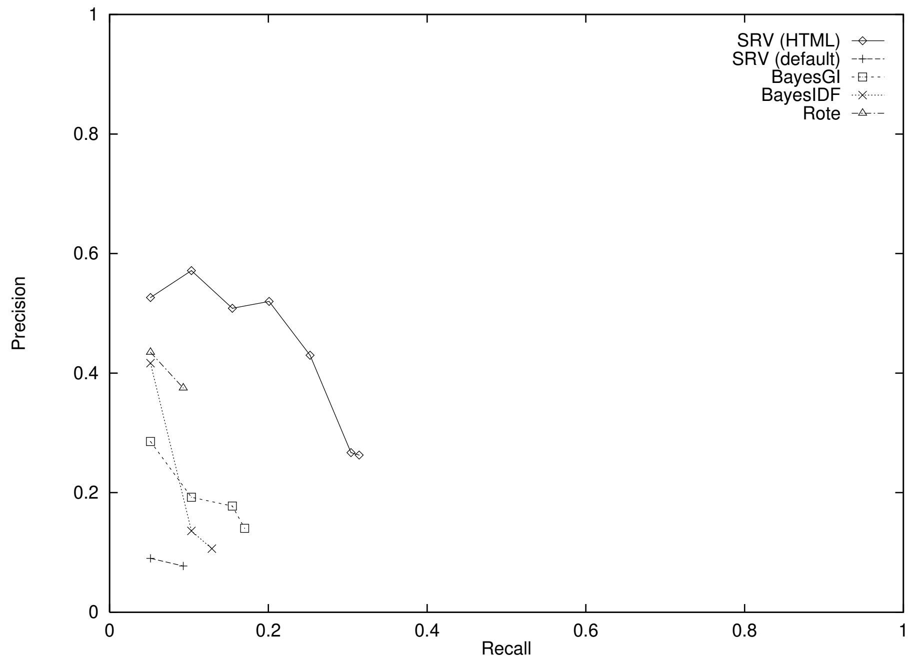

FIGURE 5.16: Precision/recall plot comparing all four learners (including SRV without HTML features) on the *projTitle* field from the WebKB domain.

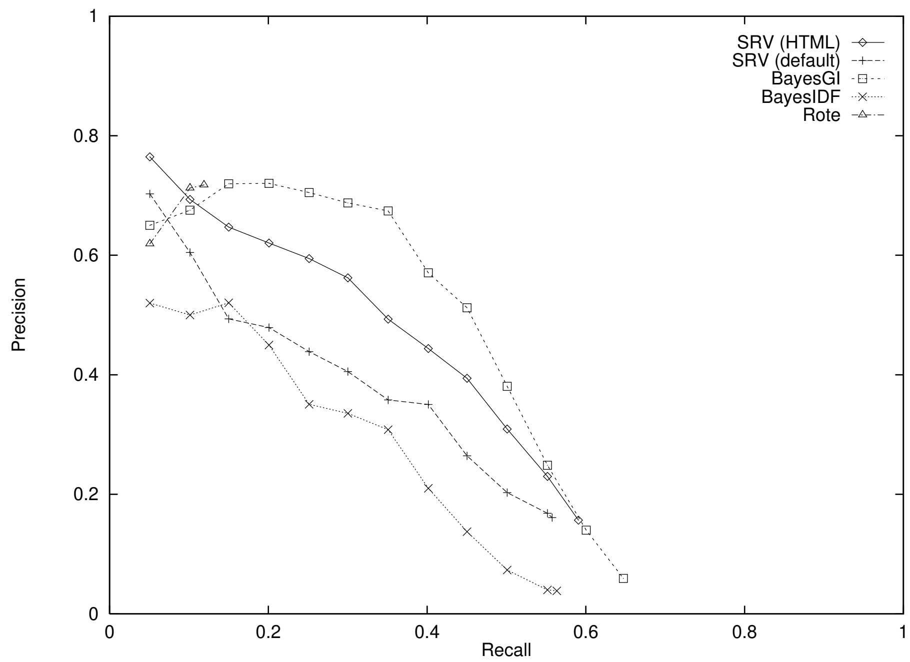

FIGURE 5.17: Precision/recall plot comparing all four learners (including SRV without HTML features) on the *crsInst* field (a "many-per-document" field) from the WebKB domain.

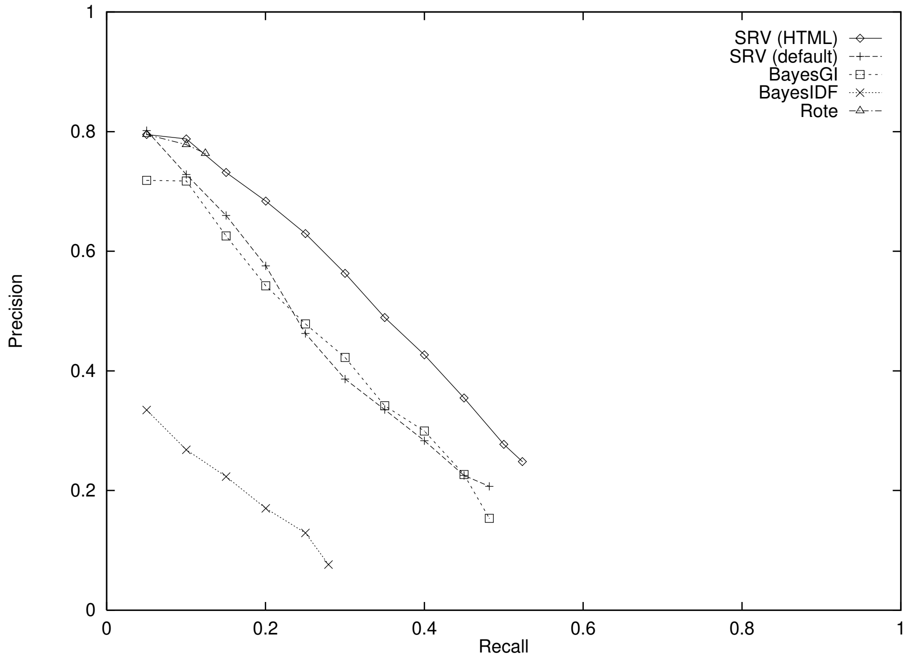

FIGURE 5.18: Precision/recall plot comparing all four learners (including SRV without HTML features) on the *projMember* field (a "many-per-document" field") from the WebKB domain.

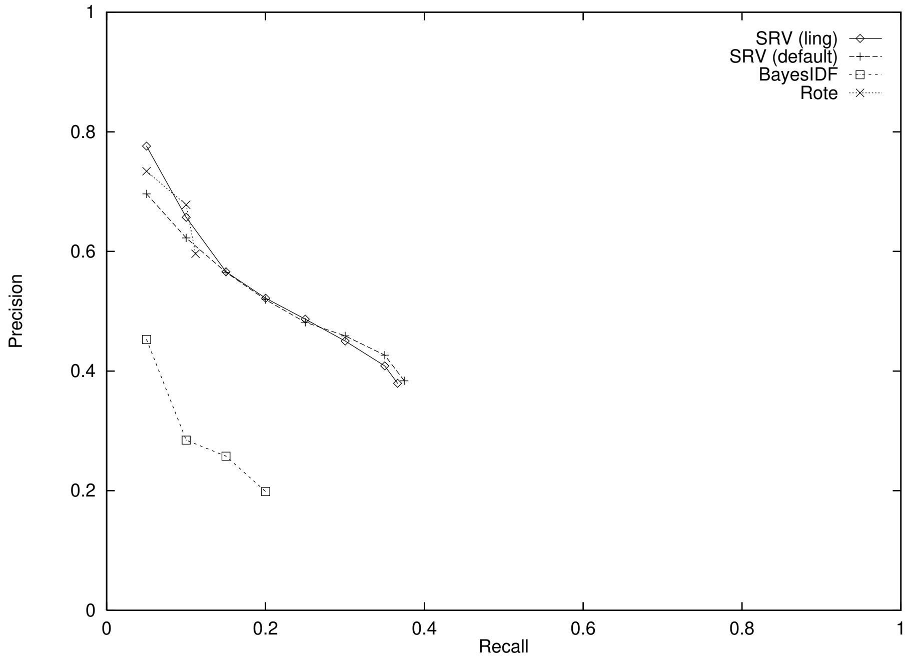

FIGURE 5.19: Precision/recall plot comparing all four learners on the *acquired* field for the acquisitions domain.

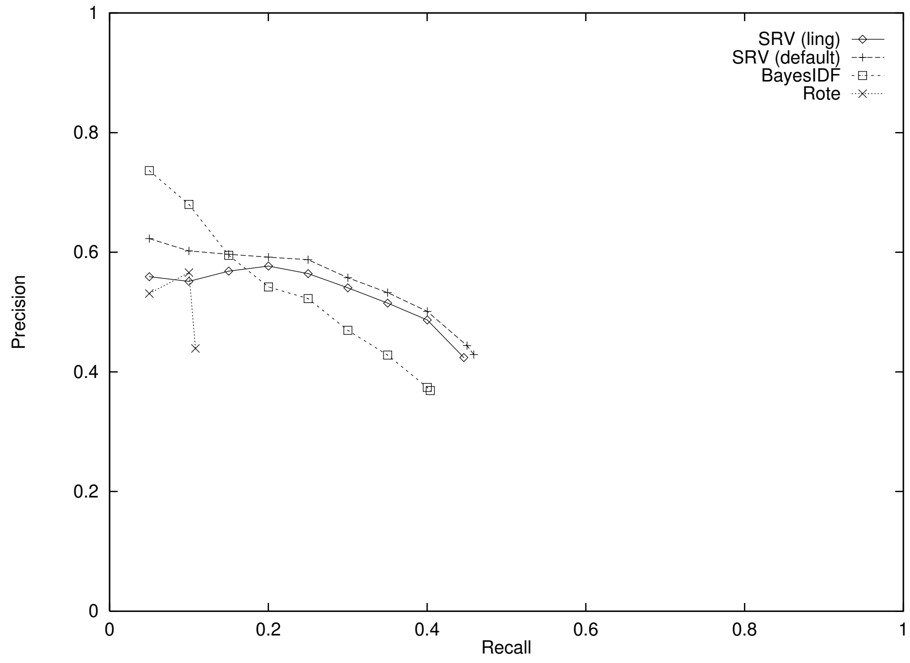

FIGURE 5.20: Precision/recall plot comparing all four learners on the *purchaser* field for the acquisitions domain.

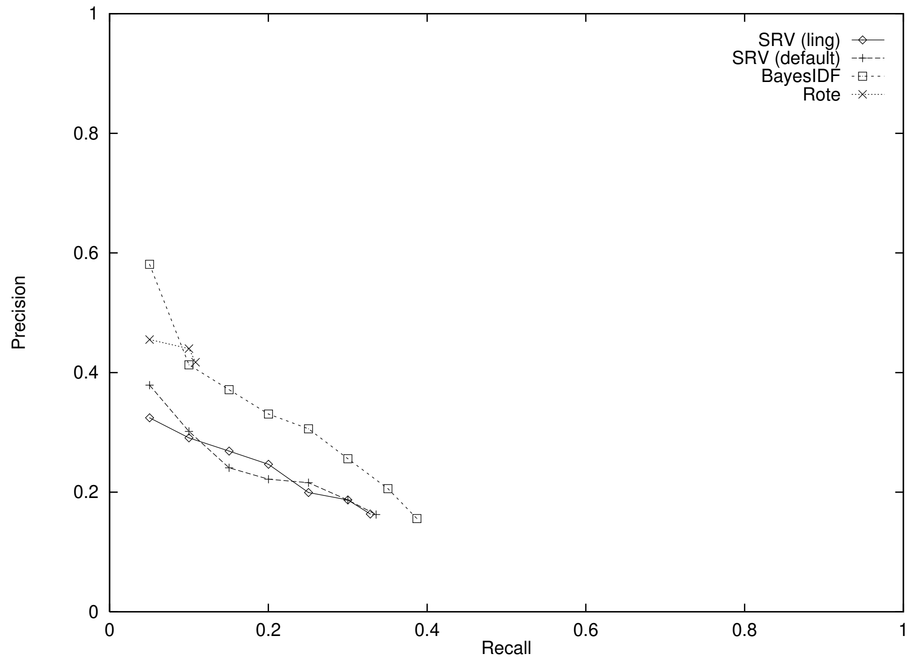

FIGURE 5.21: Precision/recall plot comparing all four learners on the *seller* field for the acquisitions domain.

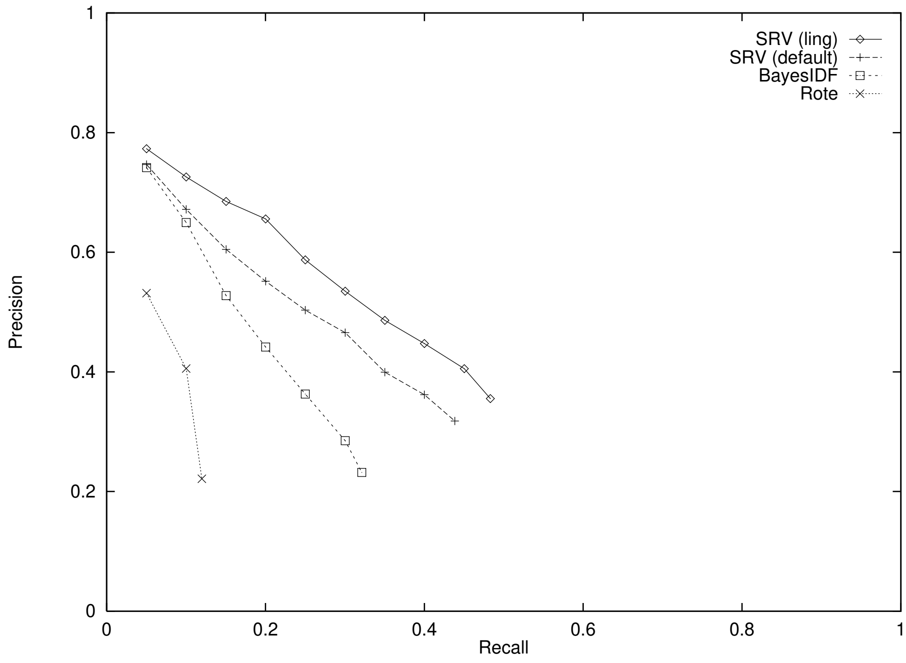

FIGURE 5.22: Precision/recall plot comparing all four learners on the *acqabr* field for the acquisitions domain.

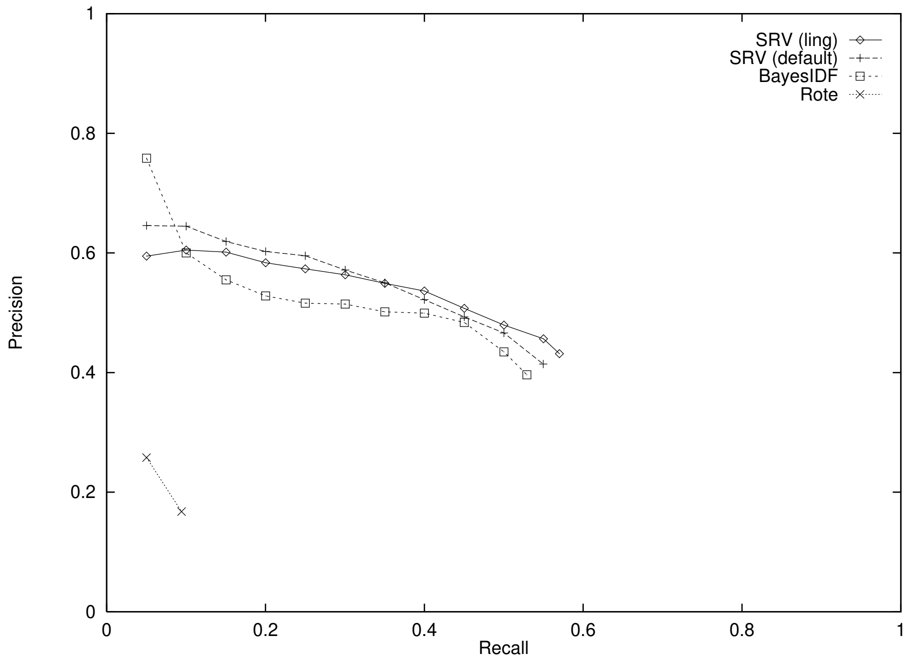

FIGURE 5.23: Precision/recall plot comparing all four learners on the *purchabr* field for the acquisitions domain.

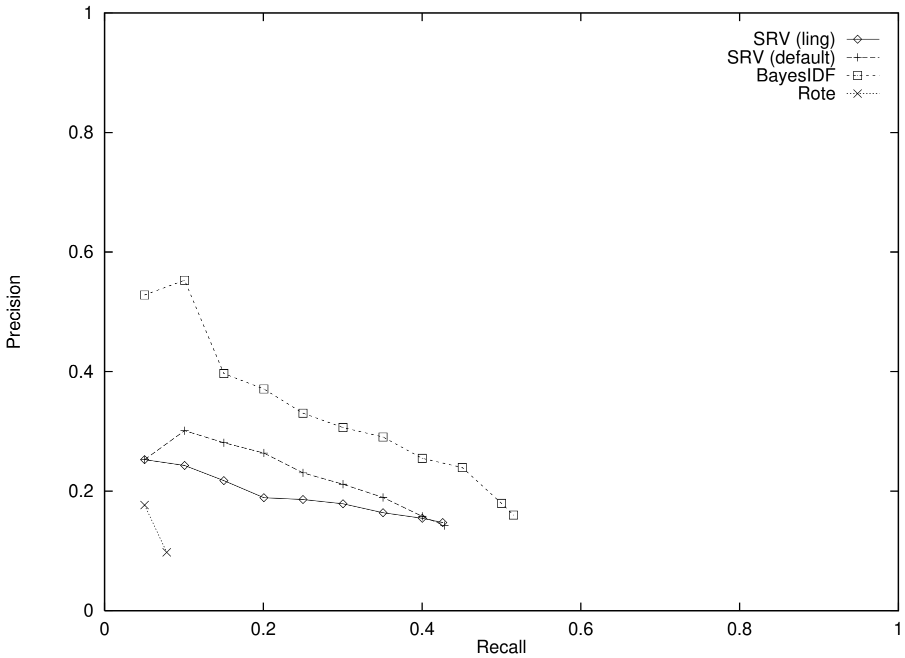

FIGURE 5.24: Precision/recall plot comparing all four learners on the *sellerabr* field for the acquisitions domain.

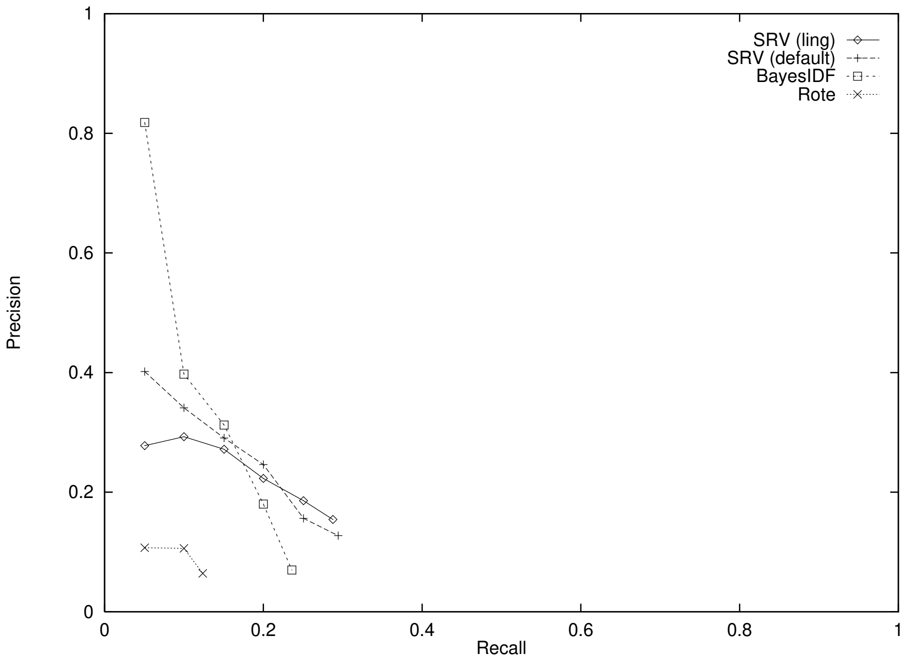

FIGURE 5.25: Precision/recall plot comparing all four learners on the *acqloc* field for the acquisitions domain.

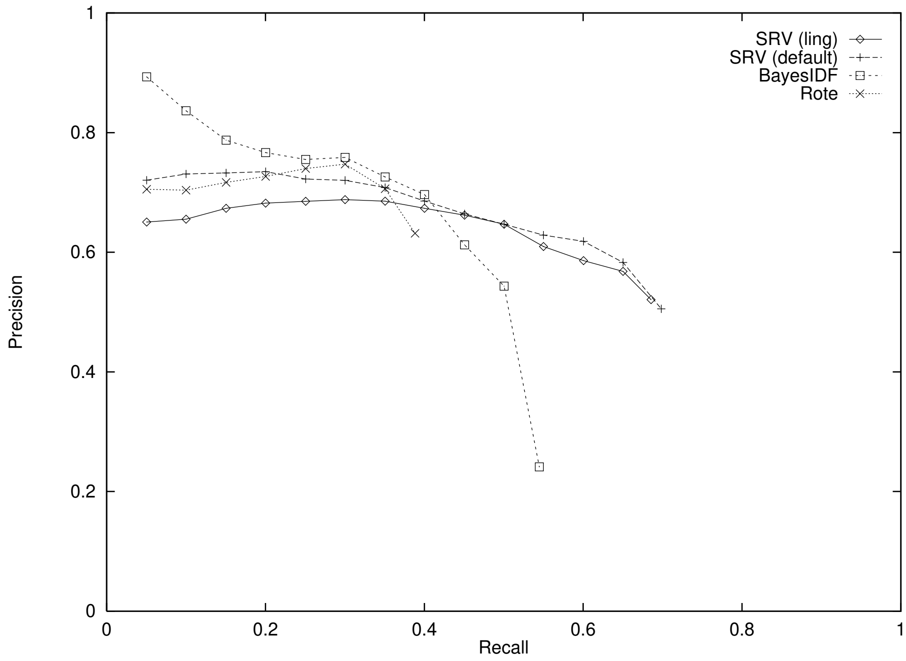

FIGURE 5.26: Precision/recall plot comparing all four learners on the *dlramt* field for the acquisitions domain.

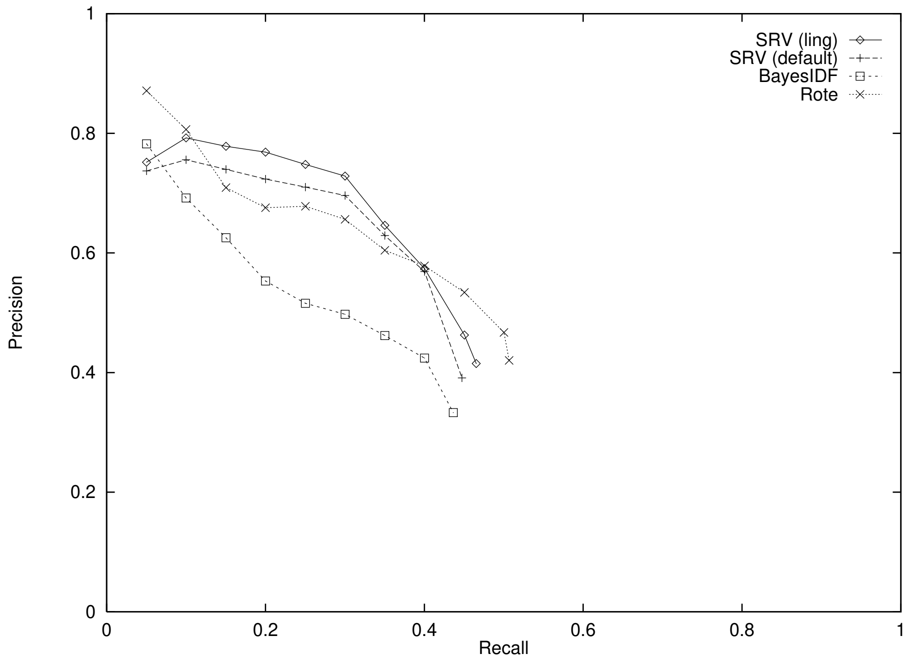

FIGURE 5.27: Precision/recall plot comparing all four learners on the *status* field for the acquisitions domain.

Chapter 6

Multistrategy Approaches

In only a few of the problems explored so far does any single learner reach very high performance levels. Each individual learner is limited, among other things, by the representational choices it embodies. This chapter asks whether these limitations complement each other and, if so, how a system working with *all* learners might benefit. A framework is presented for combining the learners without reference to the details of their implementations. For each learner, the relationship between the confidence of a prediction and the probability that it is correct is modeled using regression. This chapter presents two ways of building these regression models and three ways of combining learners' predictions. The best of these three combining methods almost always (except on one field of the fourteen tested) yields better precision and higher recall than the best of the individual learners.

The experimental results presented in the preceding chapter lead, I think, to two basic conclusions:

- All things considered, SRV is the best of the four learners. Its approach of disjunctively learning multiple sub-patterns fits the nature of the problem well, and the flexibility with which it handles abstract features makes it both powerful and convenient.

- Strictly speaking, however, there is no single best learner. SRV does not perform best on all fields. Even Rote outperforms competing learners in some cases.

Something like SRV, therefore, is an attractive candidate if a single algorithm is sought for a set of information extraction problems. Ideally, however, choosing a learner should depend on the individual task.

	Rote	BayesIDF	BayesGI	SRV
Rote	1	0.22	0.12	0.09
BayesIDF	0.81	1	0.40	0.30
BayesGI	0.93	0.85	1	0.66
SRV	0.81	0.68	0.72	1

TABLE 6.1: Outcome contingency table for the *speaker* field showing the probability that a row learner handled a document correctly, given that a column learner handled it correctly.

6.1 Opportunity

But is the best we can hope for—to choose a single, most appropriate learner for each task? As this chapter will show, the answer is "no." Even if SRV performed best on all fields, there might still be room for improvement by using "weaker" learners to compensate for SRV's blind spots. Consider the performance of Rote on the *location* field. Although Rote does not achieve the recall levels reached by other learners on this field, there is some subset of *location* instances for which it is to be trusted over other learners. The question is how to know when to trust it.

The idea presented in this chapter is to *use the confidence of a learner in deciding whether or not to trust it*. Provided that a learner is well-behaved—provided that high confidence means greater probability that a prediction is correct—we should lend greater credence to a learner's high confidence predictions. If Rote predicts with high confidence that a fragment is a *location*, we probably should accept the prediction, even over conflicting predictions from SRV.

The basic approach followed in this chapter is to gather predictions from all learners for a particular document and decide which prediction is best by looking at the confidences associated with the predictions. Of course, in order for this to work we need the learners to complement each other. In other words, do the top-performing approaches for a given field show the same behavior on a document-by-document basis? Is it the case that they find a common performance ceiling, beyond which it is hard to climb with the available features? Does the best learner simply subsume the others, predicting correctly on all documents for which they predict correctly? If so, then the best we can probably do is to select the most consistent learner. If, on the other hand, individual learners solve different parts of the problem space, then there is hope for performance superior to that achieved by any single learner.

Table 6.1 suggests that, in the case of the *speaker* field, such improvement is indeed possible. In the table, if R is the row learner and C is the column learner, then the entry in (R, C) is the probability that R handled a document correctly, *given* that C handled it correctly. It shows a lack of agreement among learners on a surprisingly high fraction of documents. Performance numbers shown for SRV and BayesGI may leave the misleading impression that their document-by-document behavior is similar. This table shows that

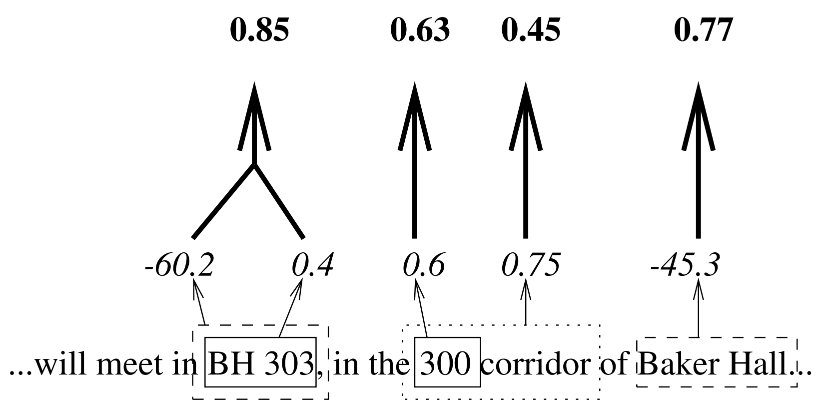

FIGURE 6.1: Combining predictions of different learners for a hypothetical *location* fragment.

it clearly is not. In fact, of the 757 documents for which at least one of these learners predicted correctly, only 399 were handled correctly by both. If we possessed an oracle for this field which, in every case where the two learners disagreed, told us which of the two to trust, we could achieve 68% precision at 73% recall. Compare this with the performance of SRV, which manages 54% precision at 58% recall. And surprisingly, the oracular figure is not necessarily an upper bound. Both SRV and BayesGI make predictions which have confidence levels that are too low to figure in the results. In cases where top predictions from both learners disagree and are both wrong, they may be issuing predictions of lower confidence which agree on the correct fragment. Thus, although both individual learners may be wrong, it may be possible nevertheless to combine them for a correct prediction. Figure 6.1 depicts this graphically. Each box style in the figure is intended to represent a different learner. By combining evidence from multiple learners (e.g., the predictions made for "`BH 303`"), we can compensate for the mistakes of individual learners.

6.2 Combining Methods

Given a document, how then should we decide among predictions returned by diverse learners? This section presents one way of making this decision. Regression over predictions made by learners on a validation set is used to map confidences to probabilities, i.e., to normalize them into the range $[0, 1]$. This makes it possible to compare predictions made by different learners directly. The interesting question is what to do with two predictions that agree, that refer to the same fragment. How should we compute the probability to assign to such a fragment based on the estimates we calculate for the individual predictions? This section three different methods for making this computation.

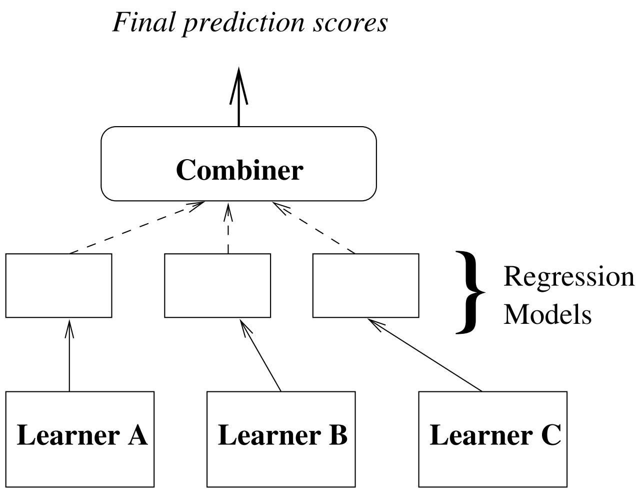

FIGURE 6.2: The basic combination scheme.

6.2.1 Basic Scheme

We cannot assume that prediction confidences bear any resemblance to true probability of correctness, or that they are comparable across learners. For example, while Rote's predictions *do* appear to be good estimates, BayesIDF's confidences are large negative log probabilities. We *do* assume, however, that probability of correctness increases with increasing confidence for all learners. The basic idea, therefore, is to attempt to compute a mapping for each learner from confidence to probability of correctness. Figure 6.2 shows this in outline. Regression models based on learner performance on hold-out sets are used to map raw confidence scores to probabilities. The combiner uses these probabilities to order all predictions. The specific steps involved in this procedure are:

1. **Validate performance on a hold-out set.** Reserve a part of the training set for validation. After training each learner, store its predictions, with confidences, on the hold-out set.

2. **Use regression to map confidences to probabilities.** Based on the learner's performance on the hold-out set, attempt to model how its performance varies with confidence. What is modeled, and the kind of regression used, depends on the combination method.

3. **Use the regression models and calculated probabilities to make the best choice on the test set.** Re-train all the learners on the full training set. Given the re-trained learners and the corresponding models, decide for each document which is best of all the fragments selected by any of the learners.

I have experimented with three basic methods of combination. The first two, which I will call CMax and CProb, both attempt to work with regression models that map directly from

```
Function CProbHandleDocument(doc)
   allpreds = All predictions made by all learners for doc
   scores = Empty hash table
   For pred in allpreds
      frag = Fragment the prediction identifies
      conf = Original confidence of pred
      learner = Learner that issued pred
      model = learner's regression model
      est = MapConfidenceToEstimate(model, conf)
       If frag is not in scores
          scores{frag} = 0
       End If
       scores{frag} = 1 - (1 - scores{frag})(1 - est)
    End For
    Sort fragments in scores best to worst
    Return Sorted fragments
 End Function
```

TABLE 6.2: The function used by CProb to process a test document.

confidence to probability of correctness. The third, which I will call CBayes, uses Bayes' Rule to make combination decisions.

6.2.2 Regression to Estimate Correctness

If a learner's confidence numbers are meaningful, then the probability that a prediction is correct will increase with increasing confidence. CMax and CProb use linear regression to model the rate at which this probability increases. For a given learner, the range between the lowest and highest confidence predictions seen on the validation set is divided into 10 intervals of equal size. Predictions are then distributed into 10 bins, depending on which interval they fall into. Once all predictions have been collected from a learner, the bins are used to generate ten datapoints (x, y) for regression, where x is the midpoint of an interval and y is the ratio in the corresponding bin of number of correct predictions to total number of predictions. Linear regression over these datapoints then completes the modeling step.

Given a field, the regression step yields one line equation for each individual learner. The confidence of any prediction by a learner on a test fragment is now converted to a probability estimate by means of the respective equation. If a test fragment has drawn only a single learner's prediction, then the output confidence is simply this estimate. The interesting question is what to do with predictions from multiple learners for a single fragment.

I experiment with two methods that are based on these linear regression models. One method,CMax simply takes the largest estimate as the official one. Given P_i, the estimates computed for each of the predictions on a fragment , CMax's combined estimate is $\max_i P_i$. In contrast, the other method, CProb, is predicated on the assumption that the fact that two or more learners agree on a prediction provides more information than either prediction alone. If we assume that two probability estimates of an event, P_a and P_b, are independent, then the combined probability is the probability that they are not both wrong, i.e., $1 - (1 - P_a)(1 - P_b)$. CProb's estimate is based on this assumption. Given a set of probability estimates P_i, its estimate for the combined probability is $1 - \prod_i (1 - P_i)$. Table 6.2 presents pseudocode for CProb's document handling procedure. The procedure for CMax is identical, except for line 13. In CMax, this line would assign to the hash entry the maximum of the current entry and the computed estimate.

6.2.3 Bayesian Prediction Combination

Although CProb may exploit the availability of predictions from multiple learners better than CMax, it still leaves something to be desired. In particular, it ignores some of the available information, such as the frequency with which a learner tends to predict at a given confidence level and any notion of prior probabilities.

The final combination method attempts to apply Bayes' Rule, which tells us how to maintain our probability estimates in response to incoming data. Using Bayes' Rule offers two advantages over CProb: It allows us to incorporate priors into our estimates, and it tells us how to maintain our hypothesis space so that the resulting estimates are closer to true probabilities—an advantage in terms of the accuracy-coverage trade-off.

Here, a hypothesis H_i takes the form, "*the fragment at this place in the document is a field instance*." Let $P_{Ai} = C$ be the event, *Learner A predicted fragment i is a field instance with confidence C*. For each fragment i chosen by any of the learners, we maintain two hypotheses explicitly, H_i and $\neg H_i$. Individual learner predictions $P_{Ai} = C$ are treated as events which cause us to update hypotheses. We want, therefore, to model $\Pr(P_{Ai} = C | H_i)$ and $\Pr(P_{Ai} = C | \neg H_i)$. It is more convenient, however, to model the event $P_{Ai} >= C$, i.e., the probability of a prediction with confidence *at least* C. Modeling the cumulative probability yields better statistics and allows us to avoid the arbitrary decisions inherent in binning.

CBayes uses exponential regression to model these two probabilities, i.e., we perform linear regression on pairs of the form $(x, \log(y))$, where x is a confidence level, and y is the cumulative probability of seeing a prediction for a fragment given that it either is or is not a field instance. As an example, consider the problem of creating the "positive" model $\Pr(P_{Ai} >= C | H_i)$ for some learner A. Let F be the total number of field instances in the validation set, and let $G_A(C)$ be the number of field instances identified by Learner A with predictions having confidence equal to or greater than C. For every prediction made by Learner A, we add a regression datapoint $(x, \log(y))$, where x is the confidence of the prediction and $y = G_A(x)/F$. The "negative" model $\Pr(P_{Ai} >= c | \neg H_i)$ is constructed in the same way, except over non-field-instance fragments—any fragment in the validation set identified by any of the learners. Note that exponential regression is used only because it clearly worked better than linear regression in early experiments. I have not tried any other alternatives, however, so there is ample reason to believe that the results reported for CBayes can be improved. Any modeling method that yields more accurate models for this data should lead to improved performance. In particular, logistic regression is a likely candidate.

```
Function CBayesHandleDocument(doc)
   allpreds = All predictions made by all learners for doc
   FIscores = Empty hash table
   notFIscores = Empty hash table
   For pred in allpreds
      frag = Fragment identified by pred
      learner = Learner that issued pred
      learnerPrior = LearnerPrior(learner)
      If learnerPrior > FIscores{frag}
          FIscores{frag} = learnerPrior
          notFIscores{frag} = 1 - learnerPrior
       End If
    End For
    For pred in allpreds
       frag = Fragment identified by pred
       conf = Original confidence of pred
       learner = Learner that issued pred

       /* Update field-instance hypothesis */

       FImodel = learner's FI-conditional regression model
       FIest = MapConfidenceToEstimate(FImodel, conf)
       FIscores{frag} = FIscores{frag} * FIest

       /* Update not-field-instance hypothesis */

       notFImodel = learner's not-FI-conditional regression model
       notFIest = MapConfidenceToEstimate(notFImodel, conf)
       notFIscores{frag} = notFIscores{frag} * notFIest

       /* Normalize two hypotheses */

       total = FIscores{frag} + notFIscores{frag}
       FIscores{frag} = FIscores{frag} / total
       notFIscores{frag} = notFIscores{frag} / total

    End For
    Sort fragments by score in FIscores, best to worst
    Return sorted fragments and corresponding FIscores
 End Function
```

TABLE 6.3: The function used by CBayes to process a test document.

With each prediction, we use the two models associated with a learner to adjust the posterior probabilities of the two mutually exclusive hypotheses regarding the affected fragment, always normalizing so they sum to 1. Table 6.3 shows pseudocode for this procedure. Each fragment in a test document that has been selected by at least one learner is assigned an entry in each of the hash tables `FIscores` and `notFIscores`. The number stored in this entry is the current estimate that the prediction does and does not identify a field instance, respectively. These scores are initialized with a prior that reflects our overall trust in a learner (lines 5 through 13)—the number of times the learner identified a field instance divided by the number of times the learner made any prediction on the validation set.

Processing of individual predictions is handled in the second for-loop (lines 14 through 37). In response to a particular prediction and confidence, the appropriate model is used to update the corresponding field-instance hypothesis (lines 21 through 23) and not-field-instance hypothesis (lines 27 through 29). Finally, the two hypotheses are normalized to sum to one (lines 33 through 35). As for CProb and CMax, the set of fragments are sorted according to their new estimates and returned.

6.3 Experiments

I experimented with the three combining methods on 14 fields from the three domains. With each field and for each learner, the modeling step involved withholding one third of the training documents for validation/regression and training the learner on the remaining two thirds. The hold-out set was fixed for all learners and all fields in a given domain. All four learners we have introduced in this thesis—Rote, BayesIDF, BayesGI, and SRV—were used as input to each combining method. For experiments with the WebKB fields, SRV was provided with the HTML-specific features which earlier experiments showed improve its performance in that domain. For the acquisitions fields SRV used its default feature set, but in contrast with the experiments in this domain presented in Chapter 5, it also had access to the every-predicate. This boosted its performance on all acquisitions fields.

Table 6.4 compares the peak F1 score achieved by each method with the score of the best individual learner according to this metric. The scores in bold face have a slightly different meaning than in previous F1 comparisons. In each case, the best individual learner is compared with the best combining method. If the better of the two is indeed better with 95% confidence, it is shown in bold. With a few exceptions, all combining methods yield better performance than the best individual learner. In only once case (*crsNumber*) does the best individual learner (BayesGI) outperform all combining methods, and the difference between its performance and that of the best combining methods for this field (CBayes) is statistically insignificant.

Note that this is also the only field on which any individual learner outperforms CProb, which is the most consistent of the three combining methods. In some cases, the improvement returned by CProb is marginal, in others it is quite substantial. On a couple of fields, it returns approximately 8 points improvement. Recall that the F1 measure represents a

	speaker	location	stime	etime	
Best individual	58.3	73.7	98.5	94.0	
CMax	61.5	79.3	97.3	92.4	
CProb	**66.2**	**79.7**	**99.3**	94.3	
CBayes	65.9	76.2	98.5	90.7	
	acquired	purchaser	acqabr	dlramt	status
Best individual	40.7	47.9	39.6	60.8*	50.9**
CMax	43.7	49.1	38.9	61.8	52.0
CProb	45.6	**53.0**	**43.1**	**64.3**	58.6
CBayes	**45.8**	49.8	41.1	61.6	**59.5**
	crsNumber	crsTitle	projTitle	crsInst	projMember
Best individual	89.9*	55.9	33.7	48.1*	41.1
CMax	87.9	58.2	33.8	44.2	43.0
CProb	88.9	62.0	34.1	49.8	45.5
CBayes	89.4	**64.5**	34.1	**52.6**	**47.2**

TABLE 6.4: Peak F1 scores of the multistrategy approach compared with that of the best individual learner (SRV in all cases, unless marked (*) for BayesGI or (**) for Rote).

mean of precision and recall. Thus, if a baseline learner achieves 50% precision and 50% recall, then an 8-point jump in the F1 score might correspond to 58% precision *and* 58% recall. If there is no change in precision, this same jump represents an improvement to 69% recall. In either case, a substantial reduction in error has been achieved.

Although CMax is clearly an adequate approach, and although it manages to outperform the best individual learner on 9 of the 14 fields, it is the worst of the three approaches. CBayes, on the other hand, is a little harder to assess. True, it performs worse than the best individual learner on more fields than CProb. Still, on the WebKB fields it appears to be the method of preference, yielding the highest performance on all five WebKB fields. On one field (projTitle), it improves over the best individual learner by almost 9 points.

It is perhaps not too difficult to say why the performance of CBayes is best on these particular fields, though exploiting the insight *is* difficult. The most salient difference between the WebKB fields and the others is their data sparsity. For each such field a learner is given access to approximately 50 documents, in contrast with 240 and 300 documents for the seminar announcement and acquisitions fields, respectively. One third of these documents are used for regression—which leads to a comparatively small number of data points. It appears that assumptions implicit in the particular kinds of regression performed as part of CBayes are more appropriate for smaller data sets, i.e., that more data hurts the performance of CBayes. Obviously, the details of regression, as used for all combining methods, need closer inspection. By the same token, these inconsistencies give us reason to believe that, with more appropriate model construction, the performance of CProb or CBayes, or both, will improve.

Figures at the end of the chapter show the full precision/recall graphs for all fields—Figures 6.3 through 6.6 for the seminar announcement fields; Figures 6.7 through 6.11 for the acquisitions fields; and Figures 6.12 through 6.16 for the WebKB fields. Examination of these graphs corroborate the insight that regression stands to be improved. In several cases, although combining methods yield a general improvement in precision and recall, their high-precision performance is worse than that of the best individual learner. A striking example is the graph for crsTitle (Figure 6.13. That the high-confidence predictions of the combining methods fare worse than those of SRV on this field indicates that, for whatever reason, they are placing undue trust in the predictions of learners that are worse than SRV. This can only mean that regression models for the various learners are faulty, and that better models in all likelihood will lead to better performance.

6.4 Discussion

In spite of the flaws, these experiments are a success. They show that it is consistently possible to find more instances of a field, and to make fewer errors in the process, by attacking the problem with a heterogeneous set of learning algorithms. It would be desirable to apply this general idea to problems other than information extraction. This section discusses the factors peculiar to information extraction that contribute to the success of these methods and speculates about their application to other tasks. I have argued in favor of diverse *representations* for information extraction, but have pursued solutions that involve diverse *learners*. The final part of this section distinguishes these two concepts and discusses their connection.

6.4.1 Favorable Factors

What, then, are the factors associated with information extraction that contribute to the success of this multistrategy approach? There are several, among them the following:

- **Examples have multiple representations.** Because documents and text fragments are "natural" objects which must be mapped to appropriate representations for learning, multiple mappings are possible. Although some information is necessarily lost in any one mapping, we can hope that taking multiple views of a document will permit better overall performance. Thus, to the extent that a problem has been "denatured," that the representation of objects has been fixed, we might expect that value of a multistrategy approach to decrease. This is not to suggest that multistrategy learning will not work for such problems; it is simply to say that the opportunity afforded by a natural object for a large number of views with greater complementarity leaves more room for improvement by means of multistrategy methods.

- **The problem is essentially Boolean.** Performing extraction can be reduced to the task of accepting or rejecting candidate text fragments. Consequently, we can gauge a learner's performance on validation documents in an attempt to model the relationship between prediction confidence and probability of correctness.

- **Each document is a case study.** In contrast with a traditional classification problem, each performance unit, a document, is a collection of test problems. *Overgeneration*, the problem of saying *yes* to too many text fragments, can be regarded as an asset when multiple learners are available. It both affords more data for our attempt to model a learner's usefulness, and holds forward the hope that the poor predictions of a single learner can be corrected by checking them against those of other learners.

Of these three conditions, perhaps only the first one is necessary. The other two conditions enhance the applicability of this approach, but their absence does not mean it will not work. To try this approach, it is sufficient to have a reasonably heterogeneous set of learners—at least two—each of which is implemented to return a confidence with each prediction. Thus, a good target for further experiments along these lines is the problem of document classification, where state-of-the-art learning approaches typically make predictions based on the values of real-valued functions. And because document classification works with the same kinds of objects as information extraction, we can hope that the idea of taking multiple views in classification will lead to better performance there, as well.

6.4.2 Representation and Learning Paradigm

It is intuitively appealing to consider the various ways in which a document can be represented, and to imagine how these views might be integrated for information extraction. And it seems evident that the best possible performance depends on good integration of views. In spite of all the emphasis on *representation*, however, what I have presented in this thesis is not a decomposition and integration of representations, but a set of learning algorithms from various *paradigms*.

Clearly, example representation and learning paradigm are distinct concepts. For example, BayesIDF might be enhanced so that it captures some of the typographic information, such as capitalization, that is accessible to SRV. This would amount to a change in representation, but not learning paradigm. It might be argued that, by advocating multiple representations and presenting solutions that involve multiple strategies, I have confounded two distinct ideas.

In fact, while the two ideas are distinct, they are not entirely separable. Choice of learning paradigm does constrain the set of possible representations. It rules out some representations, and renders other representational options less appropriate. If the learning paradigm is regular grammar inference, then examples must be expressed as symbol sequences. If it is propositional rule learning, then examples must be fixed-length feature vectors.

And some representational choices, while possible for an algorithm, may not be advisable. As suggested above, in addition to simple term frequency counts, BayesIDF might also count the number of times capitalized tokens or numbers occur in and around field

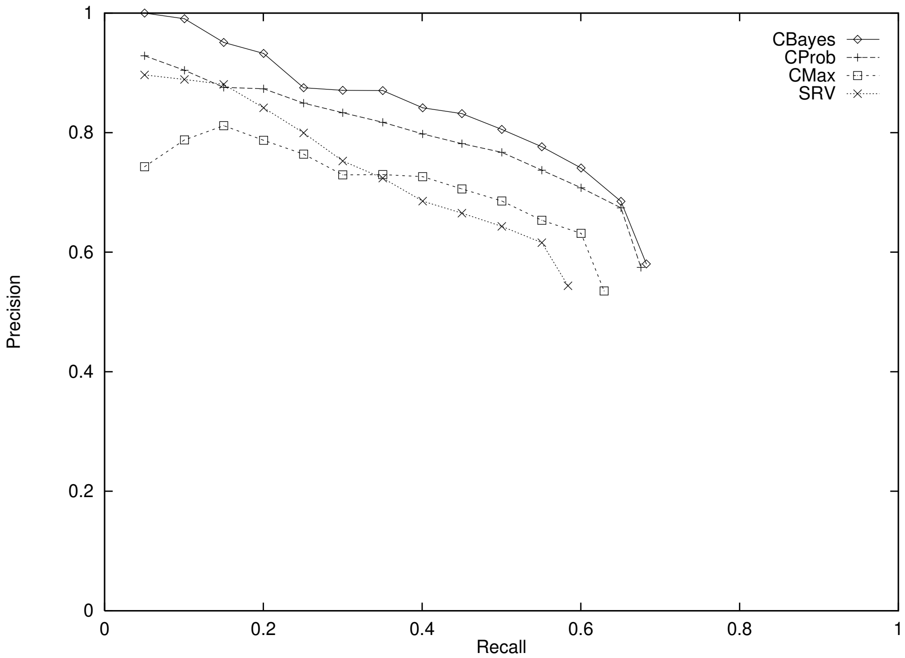

FIGURE 6.3: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *speaker* field from the seminar announcement domain.

instances, and attempt to integrate this information into its estimates. The more such features are added to its calculations, however, the greater the violation to the independence assumption. Thus, the style of learner makes certain representations less feasible.

Each of the learners embodies a number of representational commitments, some forced by the choice of paradigm, others adopted as appropriate to the learner. The view BayesIDF takes of a document may be strictly more limited than SRV's, but this very limitation makes it useful in a multistrategy setting. Rote illustrates this even more graphically. It is the least flexible of the four learners, a fact which makes its predictions very valuable for certain problems. In other words, the criteria by which we judge learning approaches in a strictly comparative setting may be insufficient for a multistrategy setting. The representation limitations of an approach, either prescribed or voluntary, may be precisely what makes it useful.

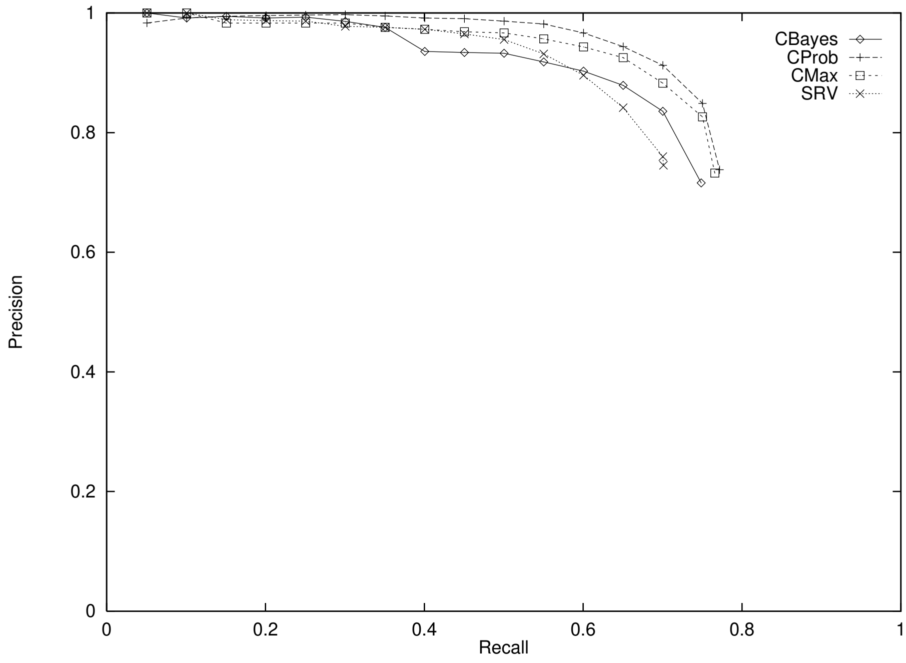

FIGURE 6.4: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *location* field from the seminar announcement domain.

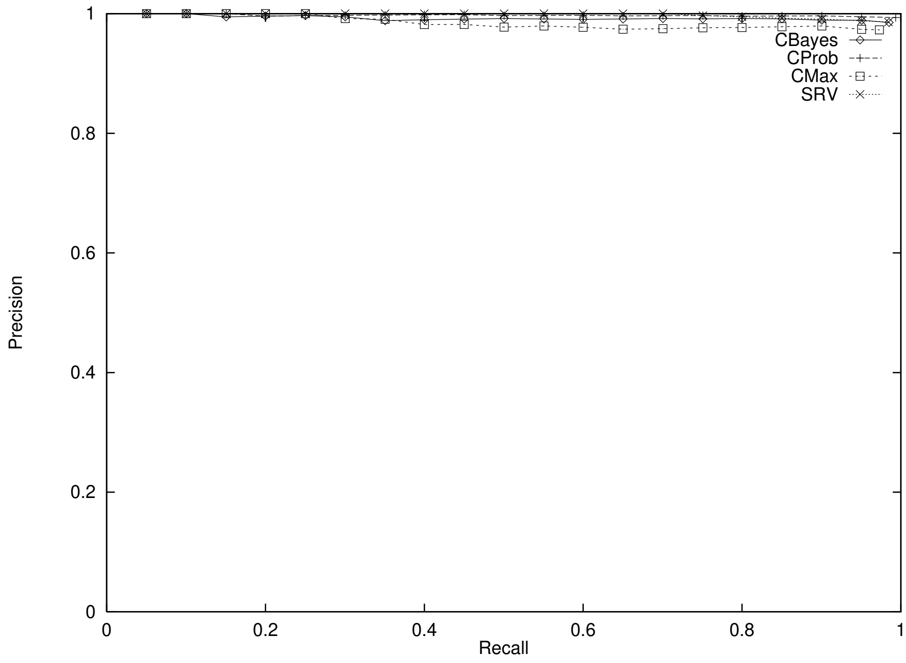

FIGURE 6.5: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *stime* field from the seminar announcement domain.

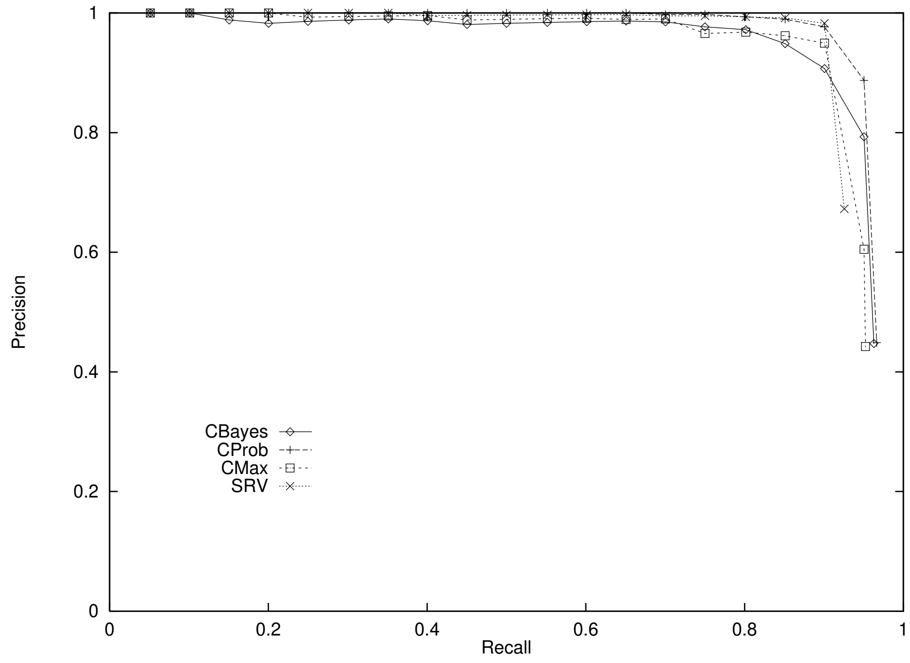

FIGURE 6.6: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *etime* field from the seminar announcement domain.

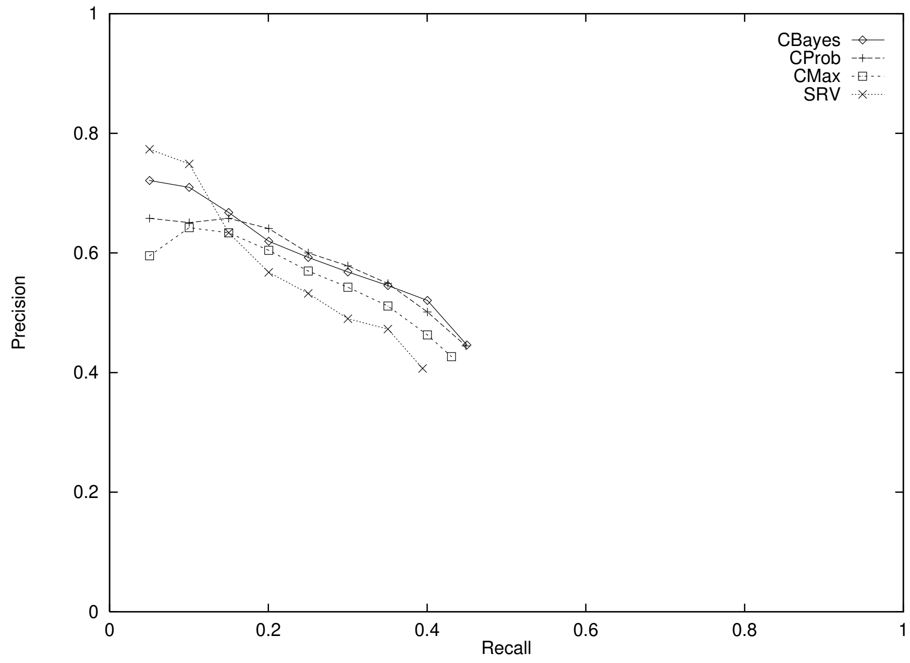

FIGURE 6.7: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *acquired* field from the acquisitions domain.

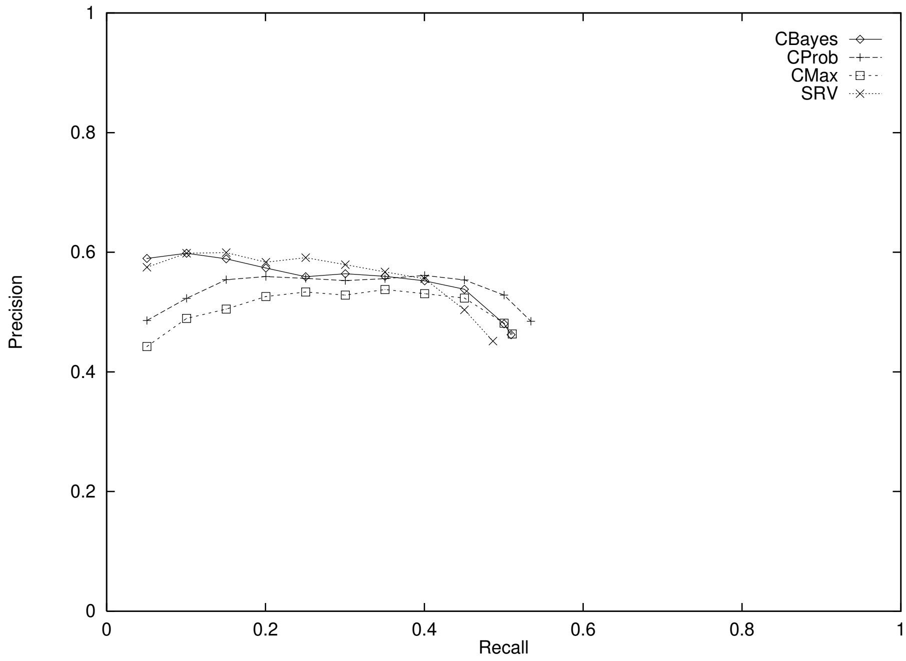

FIGURE 6.8: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *purchaser* field from the acquisitions domain.

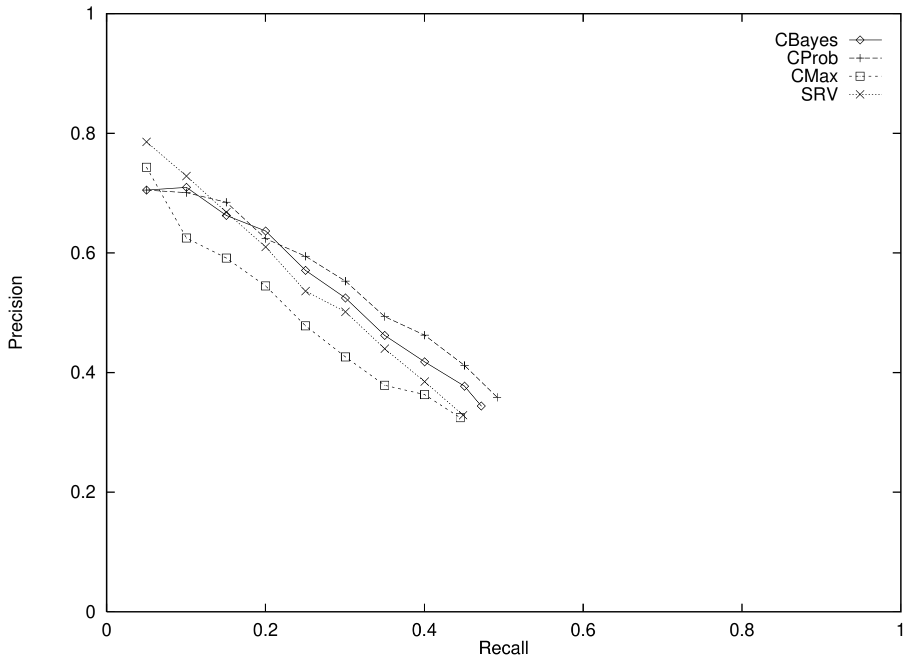

FIGURE 6.9: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *acqabr* field from the acquisitions domain.

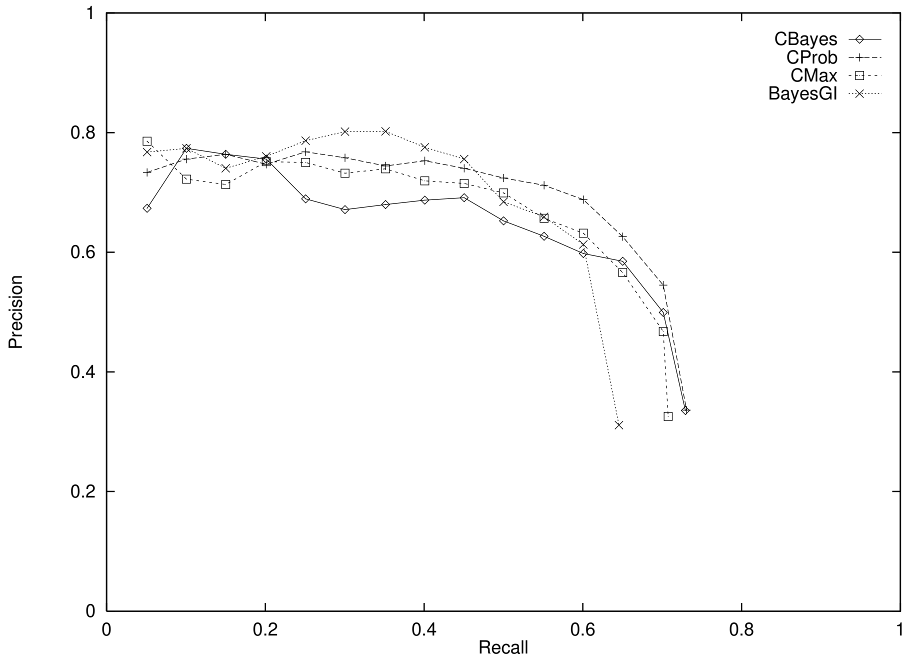

FIGURE 6.10: Precision/recall plot comparing the best individual learner (BayesGI) with the three combining methods on the *dlramt* field from the acquisitions domain.

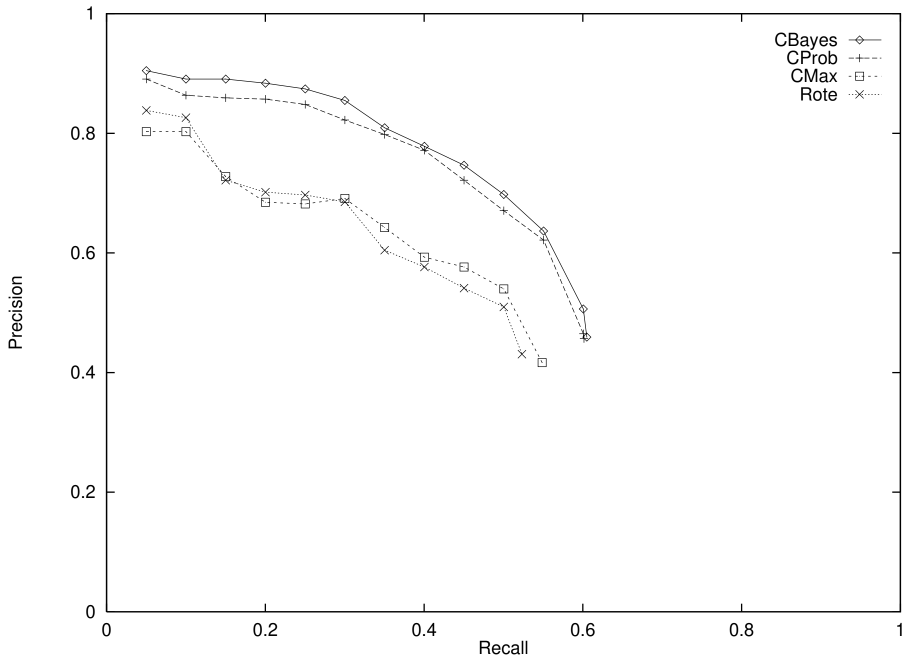

FIGURE 6.11: Precision/recall plot comparing the best individual learner (Rote) with the three combining methods on the *status* field from the acquisitions domain.

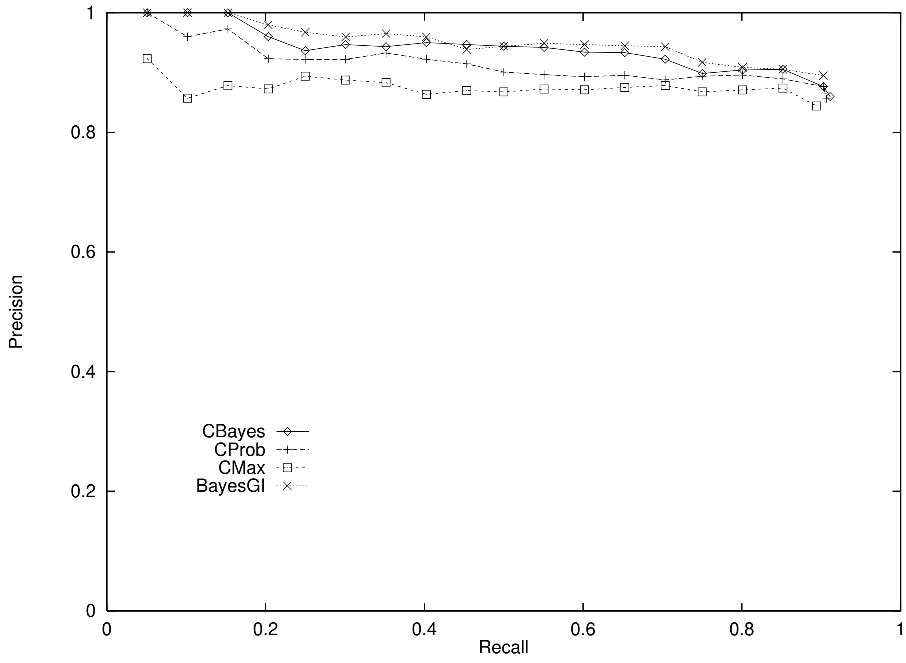

FIGURE 6.12: Precision/recall plot comparing the best individual learner (BayesGI) with the three combining methods on the *crsNumber* field from the WebKB domain.

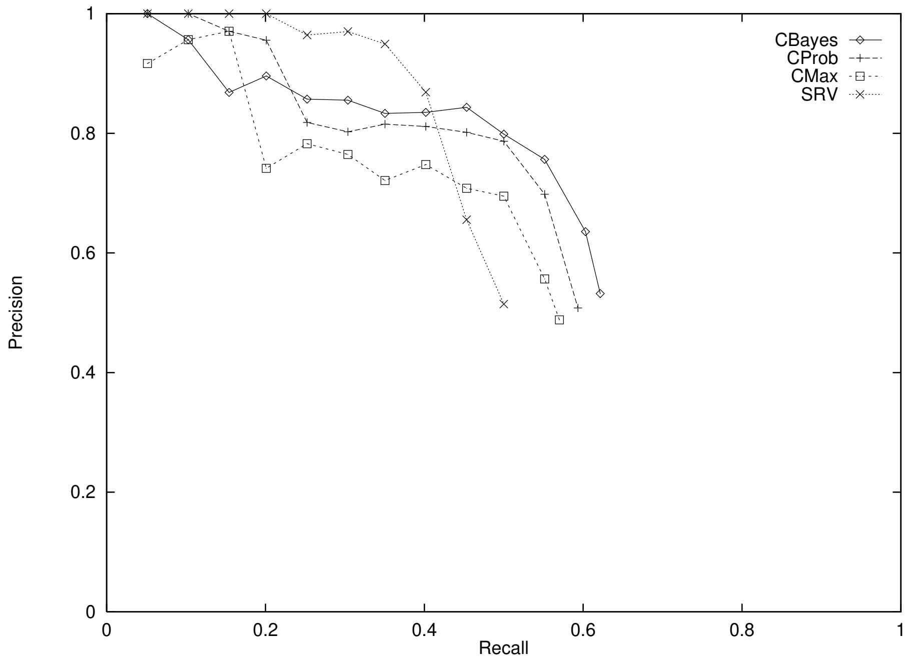

FIGURE 6.13: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *crsTitle* field from the WebKB domain.

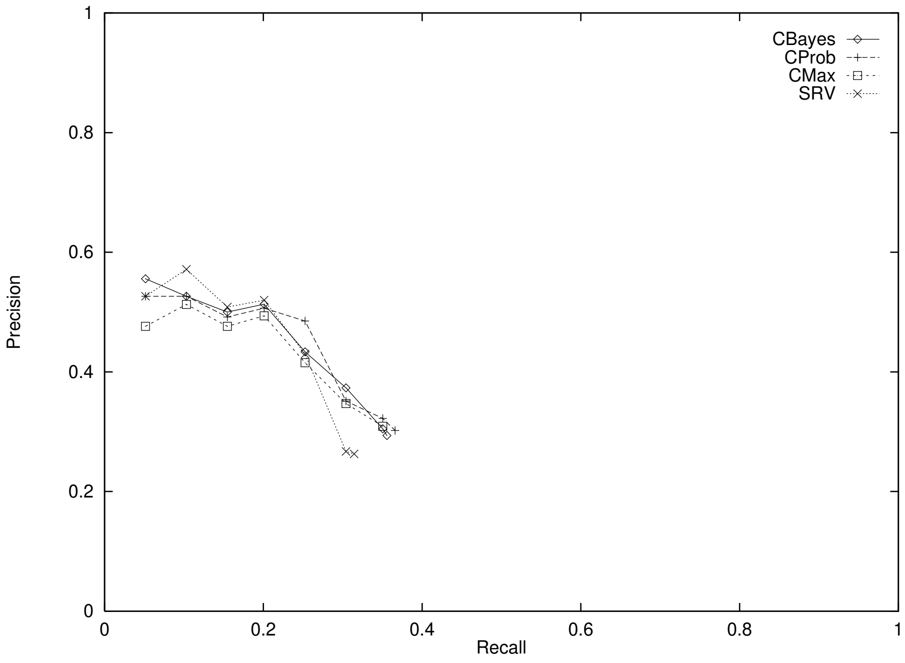

FIGURE 6.14: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *projTitle* field from the WebKB domain.

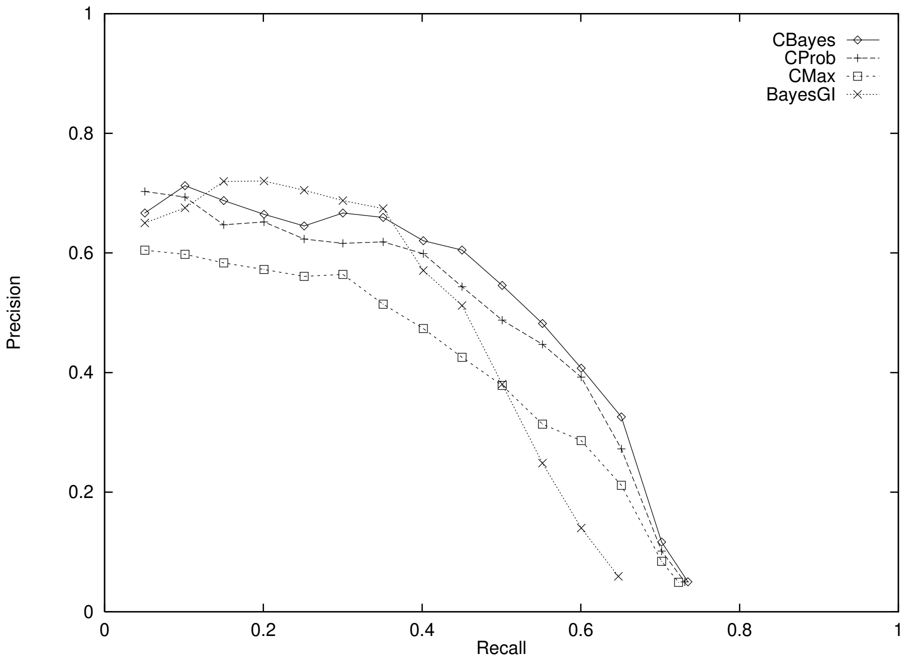

FIGURE 6.15: Precision/recall plot comparing the best individual learner (BayesGI) with the three combining methods on the *crsInst* field from the WebKB domain.

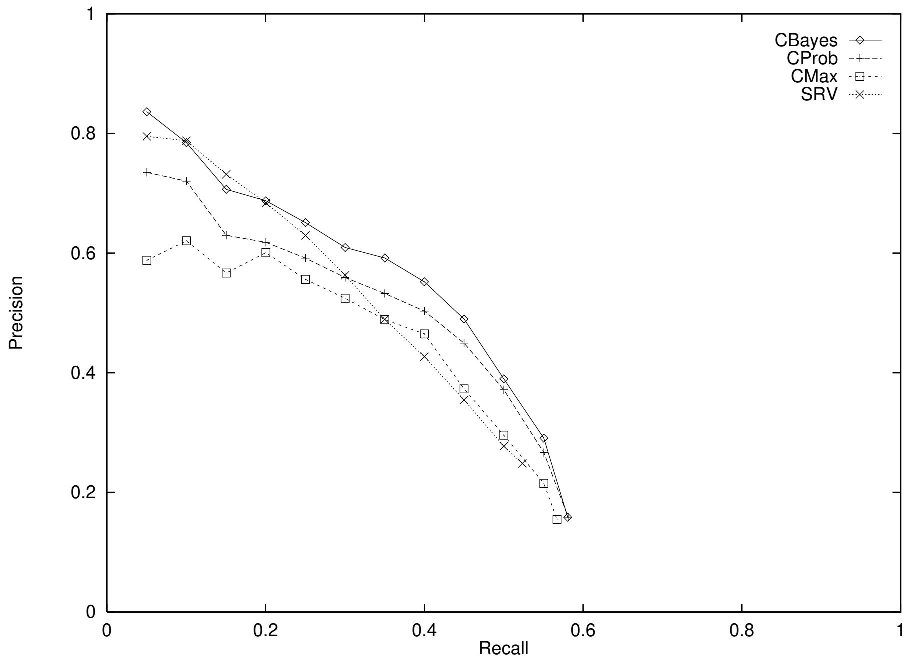

FIGURE 6.16: Precision/recall plot comparing the best individual learner (SRV) with the three combining methods on the *projMember* field from the WebKB domain.

Chapter 7

Related Work

> This chapter discusses work related to each of the individual learners presented in this thesis—the two term-space learners, BayesGI, and SRV. It attempts to cover both the relevant precedent for the general paradigm from which each learner is drawn, as well as previous applications of similar learners to the problem of information extraction. It also considers related previous work in the general problem of multistrategy learning.

The individual learning methods described in this paper are all instances—or combinations of instances—of classes of algorithms which have been investigated for some time in machine learning. And related algorithms have found application as components of various information extraction systems. This chapter is concerned with detailing both kinds of precedent, as well as pointing to related work on multistrategy learning.

7.1 Term-Space Learning

Rote and BayesIDF, as term-space learners, have many relatives in the literature on information extraction and similar disciplines. The term-space representation of a document is closely related, on the one hand, to the "bag-of-words" model, which is at the heart of much work in information retrieval and document classification; and, on the other, to representations typically used for statistical language modeling.

We can infer that something like Rote is often used as part of systems designed for MUC. An important component in many MUC systems is a domain-specific dictionary containing keywords and phrases peculiar to the target domain (e.g., (Noah and Weeks, 1993; Muraki *et al.*, 1993)). Rote can be regarded as an automated method for constructing such dictionaries, which are typically build by hand. In a MUC system, such a dictionary is typically used to permit syntactic and semantic analysis of proper nouns and jargon as part of a wider information extraction effort, rather than as a standalone slot filling component. The effectiveness of a simple memorizing learner for slot filling appears not to have been studied prior to this thesis, probably because it would have very limited applicability for the kinds of domains used as benchmarks in MUC. Rote's performance on the acquisitions domain gives us some intuition about how applicable it is for MUC-style problems.

Rote, as a machine learning approach, is a convenient baseline against which more sophisticated approaches can be compared (Mitchell, 1997). It can be regarded as a degenerate form of either k-nearest neighbor (Duda and Hart, 1973) or decision table learning (Kohavi, 1995). Naive Bayes, as an algorithm for standard feature-vector classification problems, is also often regarded as a baseline method. Unlike Rote, however, Naive Bayes can be competitive with more powerful methods (Domingos and Pazzani, 1996).

What I have called the *term-space* representation is the standard baseline approach for research in information retrieval and document classification. In these fields the *bag-of-words* model is typically employed, in which only term occurrence information, along with a statistically motivated term weighting scheme, is used. The TFIDF weighting scheme serves as the starting point for many investigations into corpus-based document management. The SMART document indexing system is prototypical of this scheme (Salton, 1971). The idea of term-space representation of text is attributable originally to Luhn (Luhn, 1958), and the idea of a Bayesian treatment for document classification to Maron (Maron, 1961). Lewis gives a thorough treatment to the use of Bayesian methods for text classification and retrieval (Lewis, 1992) and provides a large list of pointers to related research (Lewis, 1997).

In the MUC context, statistical methods have been used for various sub-tasks of the larger information extraction problem, including pre-extraction filtering (Cowie *et al.*, 1993), syntax modeling (Weischedel *et al.*, 1993), and co-reference resolution (Kehler, 1997). At least one MUC system uses Bayesian techniques to support the slot-filling task (August and Dolan, 1992). These techniques are rarely described thoroughly in the MUC proceedings, however, and the literature lacks any detailed description of them.

7.2 Grammatical Inference

Regular grammars are an appropriate formal tool for use in many tasks involving processing or classification of sequences of objects. Manually designed regular grammars have found application both for sub-tasks of the larger information extraction problem (e.g., syntax modeling (Appelt *et al.*, 1995)), and as a means of expressing extraction patterns (Evans *et al.*, 1996). Stochastic regular grammars are closely related to Markov models, which are a statistical formalism widely applied in machine learning and related disciplines (Rabiner and Juang, 1986).

Beginning with Gold (Gold, 1967), who made a theoretical survey of the space of problems and introduced some central definitions, the problem of *grammatical inference* has undergone considerable theoretical development and found diverse application (Vidal, 1994). Gold showed that the problem of finding the minimum-sized FSA accepting a finite training sample is NP-complete (Gold, 1978). This result notwithstanding, researchers have introduced methods that rely on special problem settings, such as learning by querying a teacher (Pao and III, 1978), and sub-classes of the set of regular languages (Angluin, 1982).

The term (regular) *grammatical inference* denotes approaches that seek to learn the topology of a grammar. In contrast, a Hidden Markov Model has a fixed topology, and the object is to bring model transition and emission parameters into conformance with observed data (Rabiner and Juang, 1986). An exception is the work of Stolcke and Omohundro, in which both the topology and model parameters of a Markov model are updated during training (Stolcke and Omohundro, 1994).

The general problem of grammatical inference assumes the availability of positive and negative examples, sequences from some language L and its complement $\overline{L}$, respectively. An algorithm for this setting is RPNI (Oncina and Garcia, 1992), a state-merging approach similar to Alergia (Carrasco and Oncina, 1994), which uses the negative data to prevent over-generalization. Note that the RPNI algorithm assumes no overlap between positive and negative sequences (i.e., consistency), a condition that typically does not obtain in our experiments. There are alternatives to the state-merging methodology described in this chapter. One of these is ECGI (*Error-Correcting Grammatical Inference*), which incrementally patches a grammar under construction to agree with new training data (Rulot and Vidal, 1988). In the original experiments on which this chapter is based, I compared Alergia and ECGI for this task, and obtained slightly worse performance from ECGI (Freitag, 1997).

Rivest introduces the notion of decision lists and discusses the class k-DL of decision lists having conjunctive clauses of size k at each decision (Rivest, 1987). In these terms, the algorithm used to produce alphabet transducers in this thesis outputs lists belonging to 1-DL. In broader terms, the phrase "decision list" is frequently used to describe machine learning algorithms in the covering family, such as AQ, CN2, and FOIL (Michalski, 1983; Clark and Boswell, 1991; Quinlan, 1990).

Finite state machines have been used to model linguistic syntax in at least one successful MUC system, but these were manually designed (Appelt *et al.*, 1995). Kushmerick discusses learning patterns from a sub-class of the regular languages, specifically for the purpose of performing information extraction from Internet documents (Kushmerick, 1997). His work targets a restricted class of domains involving highly regular patterns, such as Web pages automatically generated from relational databases. Wrapper-induction algorithms have since been produced that are more flexible in their handling of pattern elements and that can model embedded repeating structures (Muslea *et al.*, 1998). Closer to the work presented here is that of Goan, et al., who present a modification of Alergia designed to perform a variation of the MUC named entity recognition task (Goan *et al.*, 1996). Their algorithm allows for the integration of prior knowledge regarding character classes and can learn to recognize short, typographically distinct patterns, such as phone numbers and technical report codes. Bikel, et al., in a system they call Nymble, apply a hidden Markov model to the MUC named entity recognition task (Bikel *et al.*, 1997). The named entity problem involves extracting the names of entities belonging to some generic class. With the seminar announcements, for example, this might involve extracting the names of *all* people listed in a document, not just that of the speaker. To support learning, Nymble uses a set of simple "word features" that are quite similar to those used in these experiments. However, the feature set used is constructed so that only one of these Boolean features returns *true* for any given word. Thus, the problem of word representation and the need for transduction is circumvented manually.

7.3 Relational Learning

Rule-learning symbolic algorithms have been successfully applied to both the document classification problem (Apté *et al.*, 1994; Cohen, 1995) and NLP tasks related to information extraction (Cardie, 1993; Aone and Bennett, 1996; McCarthy and Lehnert, 1995). Early work in learning extraction patterns focused on the induction of simple rule-like features, which were then manually installed into larger information extraction systems (Riloff, 1996; Kim and Moldovan, 1995).

CRYSTAL was the first system to treat the information extraction task as a supervised learning problem in its own right (Soderland, 1996; Lehnert *et al.*, 1992). CRYSTAL is a covering algorithm, which conducts a specific-to-general (bottom-up) search for extraction rules. Rules in CRYSTAL are generalized sentence fragments. The feature set used by CRYSTAL is implicit in its search operators. It consists of literal terms, syntactic relations, and semantic noun classes. Thus, one generalization step CRYSTAL can take is to replace a literal term constraint with the semantic class to which it belongs. These semantic classes are a manually designed input to the algorithm. A potential strength of the approach taken by CRYSTAL, compared with that described here, is its ability to generate rules to extract multiple distinct field instances in concert. It is as if we set out to learn rules to recognize seminar start time and end time together. Each unique combination of slot fillers encountered together in some sentence in the training data is treated as a different learning problem. Webfoot is a modification of CRYSTAL for HTML (Soderland, 1997). Instead of sentences, Webfoot trains on text fragments that are the result of a heuristic segmentation based on HTML tags.

Rapier is closest to the approach described here (Califf and Mooney, 1997). Rapier is a specific-to-general relational learner designed to handle informal text, such as that found in Usenet job postings. In contrast with CRYSTAL, for which examples are sentences (or parsed sentence fragments), Rapier searches the less structured space of unparsed text fragments. Rapier is relational in that it in principle can exploit unbounded context surrounding field instances. It generalizes by dropping constraints from, or introducing disjunctions of constraints to, overly specific rules. Constraints are either actual terms, which must appear in the text for a rule to match, or part-of-speech labels. It appears, however, that Rapier could be adapted to exploit the kind of typographic information used by SRV.

7.4 Multistrategy Learning

The combination of BayesIDF with grammatical inference is an example of *multistrategy learning* (Michalski and Tecuci, 1994). Whereas the bulk of work in multistrategy learning has been directed at the problem of combining analytical and inductive methods, here we combine two inductive learners. This has been called "empirical multistrategy learning" (Domingos, 1996).

I have also used the term "multistrategy learning" to refer to the framework in which individual learners are used as input to a combining method, which treats them as black boxes and attempts to model their behavior in pursuit of improved performance. Similar work has been called *meta-learning* (Chan and Stolfo, 1993). One method in which diverse learners can be combined which has received considerable attention in the machine learning community is the *weighted majority* algorithm and its variants (Littlestone and Warmuth, 1994; Cesa-Bianchi *et al.*, 1997). Algorithms in this family come with theoretical guarantees that combined performance will be not much worse than the performance of the best individual learner.

Cross-validation has been shown to be an effective way to *select* from among multiple candidate learning methods (Schaffer, 1993). In the cited study, however, a single learner is selected for an entire test set, and no attempt is made to combine learner predictions for individual examples.

Multistrategy learning, i.e., combining diverse learners, is not the only answer to hard problems. Related ideas can be brought to bear when only a single learner is available. Recent years have seen the development of a family of approaches, so-called *ensemble* methods—such as bagging (Breiman, 1996), boosting (Freund and Shapire, 1996), and stacking (Wolpert, 1992)—which involve training a single learner repeatedly on different parts of a problem and combining these "specialists" for improved performance.

Chapter 8

Conclusion

> This concluding chapter summarizes the primary contributions of this thesis to research into learning for information extraction. It makes three such contributions: It tracks the performance of fixed learning approaches across several diverse domains; it compares several novel learning approaches drawn from a range of paradigms; and it shows how to combine these learners for improved extraction performance. This chapter also summarizes insights gained concerning the best use of the various learners. Finally, it identifies several promising directions for future research.

The focus of this research is the problem of information extraction: how to find a fragment of text in a document that answers a standard question. Its emphasis is on "informal" domains, domains for which linguistic processing is either infeasible or unnecessary, such as collections of Web pages or newsgroup posts. This thesis casts the problem as a choice among the many alternative fragments in a target document. I have argued that a good target function for learning is one that maps an individual text fragments to a real number—a confidence that the fragment is an instance of the target field. Extraction then boils down to eliciting the learner's output for all possible field instances in a document and selecting the one or more fragments for which the learner's confidence is greatest.

This approach—mapping fragments to real numbers—also facilitates the adaptation of standard machine learning algorithms to the information extraction problem. I have taken this as an opportunity to compare learners based on a number of learning paradigms, including statistical and term-space learning, grammatical inference, and relational learning. Different learners, I have argued, exploit different aspects of the total information available in a document. I have sought, therefore, to combine learners in order to improve performance on any given extraction problem. This thesis demonstrates one way to do this without reference to the details of any learner's implementation. Finally, because of my emphasis on informal domains, domains that violate assumptions of linguistic good form, each of the four learning algorithms described in this thesis is designed to make minimal assumptions about target documents. This makes it a simple matter to compare the learners over several different domains. Experiments reported in this thesis use three domains defined over three very different text genres: newsgroup posts, newswire articles, and Web pages.

8.1 Contributions

The contributions made by this thesis fall into four areas:

- It addresses the question, how to conduct information extraction in informal domains, domains characterized by unparsable language and dominated by non-linguistic presentational devices.
- It presents several different learning approaches and compares their effectiveness. Each of the learners is novel. Where similar learners have been used in the literature, this thesis either presents the first full explication of an approach or introduces important variations.
- It shows how to perform multistrategy learning, or learner combination, for substantially improved performance.

In this section I elaborate on each of these points in turn.

8.1.1 Informal Domains

There has been much interest recently in the problem of data mining, in both the academic and financial communities. Many of the large data repositories now in existence are collections of text. Because information extraction lies on the critical path to the successful mining of such repositories, it is only natural that the definition of information extraction has expanded to include unconventional sources. Originally, information extraction was concerned with carefully authored documents, such as newswire articles and technical manuals. Since the inception of the discipline, however, the world has accumulated a wealth of information in documents constructed under less formal conditions—in Web pages, email messages, Usenet posts.

I have operated under the assumption that natural language processing, at its current level of development, will be slow to reach competence in such genres. Note that it is not just a matter of, say, learning how to parse email message. As technology evolves, new genres of text will be born, each of which may violate assumptions made for older genres.

One of my central questions, therefore, has been, how far can we get without linguistic processing? Answering this question involved two things. On the one hand, it led to the design of learners inspired by techniques in information retrieval and statistical document classification. The result was Rote and BayesIDF, learners which operate with no more information than the occurrence patterns of raw tokens. On the other hand, it prompted me to investigate the usefulness of information that is readily accessible in English text without sophisticated processing. Two things resulted from this effort: a core set of features that captured superficial aspects of text, such as capitalization and token type (e.g., numeric, punctuation), and learners designed to exploit these features—SRV and an approach based on grammatical inference.

8.1.2 Comparison of Learning Methods

Several recent works have introduced new learning approaches for information extraction. This thesis, however, constitutes the first comparative study. Instead of a single new learner, I have introduced four new approaches and conducted a rigorous comparison. This comparison, in addition to corroborating the prevailing assumption that rule learning is as effective a paradigm as any for information extraction, turned up some surprises.

Rote learning performs surprisingly well on certain kinds of fields, and it performs best on one of the fields with which I experimented. I would argue that, given its occasionally strong performance, Rote should be a required baseline for future experiments in this area.[1] In contrast with more sophisticated learners, the workings of which can be difficult to examine, Rote is transparent. The performance of Rote can serve as one measurement of the difficulty of a specific information extraction task. And Rote is easy to implement; it embodies few subtleties that might complicate the comparison of results among researchers.

BayesIDF, too, forms a convenient baseline. Both Rote and BayesIDF make minimal assumptions about the domains on which they will be applied. And they are fast, spending only seconds on a test document. As part of conducting a comparative study of learning approaches for information extraction, therefore, I have introduced two learners that can supply useful baselines.

Measured against these baselines, the remaining two learners, BayesGI and SRV, appear superior. BayesGI, a hybrid of BayesIDF and the grammatical inference algorithm Alergia (Carrasco and Oncina, 1994), is designed specifically to correct for BayesIDF's inability to represent the appropriate structure of a field's instances. The introduction of this structural information yields substantial jumps in both precision and recall, except on those few fields on which BayesIDF achieves almost perfect performance by itself.

Clearly the best of algorithms tested, SRV is a relational learner for information extraction. Not the first such learner, SRV nevertheless differs from related work in three respects:

- It conducts the search for individual rules in the general-to-specific direction.
- It has an explicit set of features that are the building blocks of the extraction patterns it learns. This set is extensible, making it easy to introduce domain-specific information to SRV.
- It follows training with a validation step used to assign confidence scores to SRV's predictions. This makes it possible to trade precision for recall, and vice versa.

[1] One wonders how far a rote learner would get on some of the traditional MUC problems. For example, is seeing the token "FMLN" sufficient to know that the FMLN is responsible for the terrorist attack described in an article?

8.1.3 Multistrategy Learning

There are two examples of multistrategy learning in this thesis. First is the combination of BayesIDF and Alergia. The result is a learner more powerful than either constituent. While it is well accepted in the machine learning community that such combinations can lead to improved performance, this particular algorithm is a particularly striking example. Its success supports my contention that tasks involving natural objects, objects which permit several more or less orthogonal representations, are good targets for multistrategy learning.

BayesIDF and grammatical inference were chosen for these experiments because of the complementarity of their representations. The hybrid learner BayesGI is a tight, manually engineered coupling of the two constituent learners. In Chapter 6 I introduced a second, looser form of learner combination, one which treats individual learners as experts to be consulted in making a combined decision. This method is an instance of the class of voting schemes and meta-learning which are by now also regarded as tried and true techniques in the machine learning community. The application of this approach to information extraction, however, is quite novel. As with BayesGI, I show that this multistrategy approach can yield unusually large improvements. Once again, the problem of information extraction proves to be a good target for multistrategy learning. I speculate that one of the reasons for this is the fact that we can play with representations; a document is a natural object, many representations of which are possible.

8.2 Insights Gained

In addition to these major contributions, this research has afforded many minor insights. This section lists the most salient insights, particularly those into the applicability of the individual learners.

No single approach is best. The comparative results show that there is no single learner that always performs best, and the multistrategy results show that even the best learner, in any given case, usually stands to improve.

Information extraction is many problems. A correlary to the previous point is that information extraction is best regarded as a collection of related problems. This is true in a couple of senses. First, there is no guarantee that because a learner performs well on one field, it will perform well on another, even one from the same domain. What is more, a single field may embody many kinds of patterns, and the suitability of a learner may depend on which of these patterns it is able to find and exploit. Second, information extraction benefits from decomposition. BayesGI illustrates this well: Two learners, one specializing in statistical contextual patterns, the other specializing in the typographic structure of field instances, can be joined to produce a learner stronger than either individual approach.

Rote is well suited for skewed phrasal distributions. An approach like Rote is called for when field instances are highly typical of a field and atypical of text occurring outside a field. Of course, the other prerequisite is that field instances tend to be repeated verbatim in multiple documents. Thus, a field on which Rote excels is the *location* field in the seminar announcements domain. It is highly unusual to encounter room designations in this domain in any other role than as the location of a seminar, and exceptions to this rule usually correspond to rooms that are not used for seminars, such as private offices. In contrast, *stime*, the start time of a seminar, does not have this characteristic. Times are reasonably common in the seminar announcements, and they serve a number of purposes. Thus, Rote's performance on this field is much worse than that of competing learners.

BayesIDF is well suited for strong stereotypic contextual patterns. While it is generally not sufficient to recognize times in order to find instances of *stime*, start times in the seminar announcement domain are very often surrounded by strong clues. A large proportion of the start times in this domain are preceded by the phrase, "`Time:`". A preponderance of such clues allows BayesIDF to achieve near perfection on this field.

SRV and BayesGI are good for fields characterized by strong typographic patterns. Examples of such fields are *crsNumber*, the university-assigned number of a course on the course home page, and *location*, the location of a seminar listed in an announcement. University course numbers tend to follow certain patterns readily expressible in precisely the language given to these two learners. Although the details vary across universities, it nevertheless is often possible to express patterns that cross university boundaries, such as "look for a short token in upper case followed by a number."

SRV is more versatile in expressing typographic patterns. In contrast with BayesGI, SRV is not constrained to express patterns that account for all the tokens in a fragment. If the important pattern over a set of fragments involves only the first and last tokens, SRV can capture the pattern directly, while the performance of BayesGI may be hurt by the requirement that it mention all tokens.

In spite of its ability to explore unbounded context, SRV is best at *local* patterns. In contrast with BayesGI, SRV can also search for simple abstract patterns in the language surrounding a set of field instances. My experience is that in order for such patterns to be found, they must be quite local. SRV's ability to explore arbitrary contexts notwithstanding, the most important patterns are still local. In order for SRV to reach far beyond the boundaries of a field, it must first find something of value in the immediate context. And, typically, by the time a rule has extended its consideration very far into the context, its coverage is reduced to the point that finding good patterns is a challenge. Even when provided with relational features that allow large jumps (and relational features are particularly expensive), the value of such features must be greater than that of patterns to be found closer to home. As a consequence, I have not seen many high-quality rules that express non-local patterns.

None of the learners can express global patterns. None of the learners is able to say anything about the page in which it is searching or larger structures in which field instances may be embedded. A number of recent papers have reported on information extraction "wrappers," typically used with HTML. A common assumption made by such algorithms is that fields will appear as part of embedded, repeating structures. One field for which such an approach might be applicable is *projMember*, instantiations of which are the names of a project's members as listed on the official project home page. Project pages tend to present members in simple itemized lists and to include member-specific information in very regular patterns. The way in which the learning problem is formalized in this thesis cannot exploit such repeating, highly regular patterns.

8.3 Open Questions

Any thesis raises new questions while it answers old ones. A thesis like this one, the subject of which is novel and relatively unexplored, seems to raise more questions than it answers. Areas ripe for incremental improvement are hinted at in the various chapters. SRV, for example, might be strengthened in any number of ways. New predicate types might allow it to learn more effective rules. Its efficiency might be improved in several ways, leading to a more useful algorithm. Better stopping criteria and accuracy estimation could lead to small but useful increases in precision and recall. Similarly, there is much work left undone on the subject of multistrategy learning, particularly in the details of regression.

Rather than concentrate on such minor refinements, however, in this final section I want to draw attention to the holes, beginning with the small ones, those that might be patched with a few month's work, and moving to truly large open questions. I identify five open problems, and attempt to assess the magnitude of each:

- **Grammatical inference over feature vectors** Because of the sequential structure of the information extraction problem, grammatical inference is an appealing source of candidate algorithms, but assumptions underlying existing approaches fail to make use of the wealth of information available in text. Because the grammar-induction-by-state-merging paradigm is well developed and can serve as a starting point for investigations, this is probably a short-term problem.

- **Exploiting field co-occurrence** Treating the extraction of each field as a separate learning problem is both unrealistic and wasteful of available information. Some previous work shows how to exploit highly regular or local field co-occurrence, but a general approach is lacking. This item really comprises two research directions, a short-term and a medium-term one. On the one hand, given a document, one can take a learner's (or several learners') predictions for multiple fields and attempt to optimize the global extraction. On the other hand, designing learning algorithms that take into account field co-occurrence in the training phase is an interesting problem with a possibility of real impact.

- **Using linguistic information** Although SRV is sufficiently flexible to make use of syntactic and semantic information more or less directly, in the only experiments I performed in this vein providing linguistic information to SRV yielded little benefit. I suspect the reason for this is SRV's representation of the problem of information extraction, one which, to my knowledge, it shares with all other learning systems for information extraction: It assumes that information *local* to a fragment is sufficient to determine its status. Perhaps, to use linguistic information effectively, it is necessary to take a non-local view, to carry constraints between sentences. Doing this in a way that supports learning for information extraction without solving the natural language processing problem is probably a sufficiently big problem for at least one dissertation.

- **Using layout** All of the learners presented in this thesis assume that a document is a sequence of terms. SRV can exploit non-linear structure, but not the two-dimensional structure so apparent to the human eye perusing a document. I surmise that layout is very useful in many information extraction problems, particularly those involving HTML or informally constructed ASCII documents, but there is no reason to prefer any of several possible approaches as this point. Consequently, there are probably a number of interesting conference papers hiding in this problem.

- **Information extraction as navigation** A human searching for a datum in a document, particularly documents with interesting layout, quickly identifies promising regions and skips intervening text. In contrast, the approaches I have described exhaustively examine all fragments in a document from first to last. This strongly suggests a change or augmentation of problem representation might prove valuable. An interesting starting point might involve modeling the human approach to the task—probably a thesis-sized work. Failing that, perhaps the problem can be cast as one of navigation through a document, and perhaps learning for sequential decision making—reinforcement learning—can serve as a fund of useful ideas. Which ideas and how they should be applied is certainly also a good subject for a dissertation.

8.3.1 Grammatical Inference over Feature Vectors

Strongly influenced by the study of formal languages, grammatical inference generally assumes that the universe consists of sequences of symbols from some finite alphabet. Information extraction is a good target application for grammatical inference, because of its essentially sequential nature. I have argued that words, rather than, say, individual characters, form the right level of granularity at which to apply grammatical inference to this problem. But words, of course, are relatively complicated objects, having a number of relevant dimensions—morphology, semantics, orthography, typography, etc. In order to apply grammatical inference, words must be replaced by symbols from a canonical alphabet. The larger this alphabet, the greater the need for data to support effective generalization. Thus, in order for grammatical inference to be effective, all the relevant features and their values must be reduced to a comparatively small number of symbols.

My method for inferring transducers is one way in which such reduction may be conducted, but it is hardly ideal. In fact, any method which conducts the reduction *before* grammatical inference is unsatisfactory. The decision about how to represent an object with multiple features ought to be an integral part of the learning algorithm; it ought to be driven by the same criteria that drive state merging. Is it possible, however, to adapt existing grammatical inference algorithms, so that, in addition to deciding which states to merge, they also choose abstractions over the objects involved?

To make the discussion a little more concrete, let us suppose we have two seminar start times "`1:00 pm`" and "`1:30 PM`". We might modify an algorithm like Alergia (Carrasco and Oncina, 1994), in the hope that it might unify these two fragments. Recall that Alergia works by iteratively combining pairs of states it judges equivalent, and that in order to do this, it compares the emissions assigned to transitions out of the states. Suppose Alergia is considering a merge between the two states corresponding to "`00`" and "`30`" in the above fragments, and suppose the merge will be assessed as warranted only if the tokens "`pm`" and "`PM`" can be taken as equivalent. It is trivial to see that the merge is warranted, yet while the two tokens have very similar feature vectors, they are not identical. The value of a feature such as capitalized differs for the two tokens. On what basis should Alergia decide to make this merge, and how should the unified transition be represented—as the intersection of shared feature values?

Markov models are an alternative to the grammatical inference approach. Rather than learn topology, one might attempt to learn a set of weights in a grammar with fixed topology. And it should be possible to extend the standard hidden Markov model algorithms, which like grammatical inference algorithms assume observations arrive as sequences of symbols, to take into account all feature values simultaneously. Whether there is any virtue in doing this is an interesting question for future research.

8.3.2 Exploiting Field Co-occurrence

Information extraction involves mapping a document to a composite structure that represents essential aspects of its meaning. It has been convenient for me to regard this as a bundle of learning problems, one for each field in the structure. And in simple cases, in cases where each field can be extracted reliably enough, it may be sufficient to extract each field separately and assemble the structure by combining its constituents after prediction. Clearly, though, this approach is not sufficient for the general information extraction problem. What goes into each field will often be strongly influenced by other fields. If one believes a seminar will begin at 1:30 p.m., one will not consider 10:00 a.m. as a candidate end time.

It is possible to begin to take steps toward exploiting field co-occurrence with the techniques I have presented in this thesis. The best point of departure for such an endeavor is the multistrategy setting described in Chapter 6. Starting with all predictions returned by all learners for a given document, rather than decide for each field separately, it might be possible to improve performance by taking a view that considers the global effect of field assignments. If I decide, based on what my information extractors tell me, that the *purchaser* field begins the lead paragraph in an article detailing a corporate acquisition, then my expectation that *acquired* will be instantiated a few words downstream in the same sentence should be strongly increased.

Ideally, however, the co-occurrence of fields should directly inform learning for information extraction. Co-occurrence might mean proximity, as in the acquisition example in the previous paragraph, or semantic constraint, as in the start time/end time example. Serializing the extraction problems, perhaps from easiest to hardest, so that training for the harder fields can take into account predictions for the easier ones, is one step in this direction. There are problems with this approach, however. Not only might it be non-trivial to order fields, but care must be taken so that learner errors on the early fields do not hurt performance on the later ones.

It is not clear, therefore, that the learners presented in this thesis can exploit field co-occurrence without redesign. Note that at least one existing learning system for information extraction, CRYSTAL (Soderland, 1996), is designed to predict for multiple fields at once, but only when they are instantiated together within the bounds of a single sentence. And CRYSTAL treats each such combination of fields as a separate learning problem, thereby reducing the number of training examples in problems that typically are already data-sparse. Therefore, the problem of exploiting field co-occurrence is far from solved. Whether it involves simple modifications to one or more of the existing learners, or the design of entirely new algorithms is a question for future research.

8.3.3 Using Linguistic Information

My focus has been on domains characterized by "messy" text, text for which automatic linguistic processing is difficult. Consequently, I have investigated inexpensive features, features that are available in any domain and readily computable. In one set of experiments, described in Chapter 5, I attempted to integrate syntactic and lexical information into learning, in the hope of demonstrating the versatility of SRV. Results were mixed, and I was unable to conclude that SRV can make use of such information.

Some kind of linguistic information can be obtained for any information extraction domain, and any such information that is practical to obtain ought to be used. While previous work on learning for information extraction has assumed that linguistic information should be the substrate on which learning is conducted—and has gotten good results—my work has failed to demonstrate much benefit. This raises a number of questions:

- Is linguistic information really that useful for information extraction? Might it be the case that, in most domains, learning can be as effective with inexpensive non-linguistic information?
- Did the noise I introduced while obtaining linguistic information for my experiments simply outweigh any benefit SRV might have gotten from it?
- Is SRV simply ill suited to use linguistic information?

My limited experiments in this area do not give enough information to begin answering these questions. It is worthwhile to attempt to answer them, however. Obtaining good syntactic and semantic information is often difficult, particularly in the kinds of domains I have investigated, but if it means an improvement in accuracy, it is probably worth the cost. Note that there is plenty of room for improvement in the acquisitions domain. In order to make substantial headway in this domain, it may be necessary to represent the semantic content of the articles in a limited way. This may entail a completely new problem representation and new learning approaches.

Inasmuch as natural language processing is an open problem, we must content ourselves with incomplete and noisy information. One possible approach might involve recovering co-reference chains *before* training for information extraction. Previous attempts at learning for information extraction assume that co-reference resolution, if it occurs at all, is a subsequent step. It seems more appropriate, however, to express extraction patterns in terms of all the information contained in a co-reference chain—if it can be recovered reliably. And of course, a learner designed to induce such patterns would look quite different from the ones described in this dissertation.

8.3.4 Using Layout

While linguistics is an obvious source of potentially powerful information for information extraction, this thesis has not exhausted the wealth of cheap information available in many domains. One such source, which I invoked as one motivation for this thesis, is layout. In fact, there appears to be an inverse relationship between linguistics and layout. In domains that are problematic for natural language processing, layout is often used as a presentational device. Two good examples are the WebKB documents and the seminar announcements. In Web pages, layout controls are explicitly inserted into documents and are consequently available to learners like SRV. In documents belonging to the genre from which the seminar announcements are taken (email message, Usenet posts), on the other hand, layout consists of patterns of whitespace usage and alignment with document margins. It is less obvious in such a case how layout is to be used.

Yet layout appears to have large heuristic value for humans confronted with such documents. A simple experiment convinces one of this. Take a typical seminar announcement and reduce all whitespace to individual spaces, converting the text into a single block flush with the margins. Search for the seminar speaker in the original and modified documents. While one's eye can often glide directly to the desired information in the version preserving layout, one is reduced to linear scanning in the modified one.

It may be possible to make some use of layout by encoding aspects of it for a learner like SRV. Lost with such an approach, however, is any conception of larger layout structures, such as paragraphs and headers, and the ways in which these structures interact to convey meaning to the user. Seminar start times, for example, often occur in the header of an announcement. Message headers have a two-dimensional profile quite distinct from other large textual objects.[2] It is an interesting problem for pattern recognition, perhaps drawing inspiration from machine vision, to detect such larger text structures. In some cases, however, it may suffice to recognize individual text objects; it may be necessary to express patterns over multiple such objects. As in the extraction problem at the token level, the task in this case is to pick contiguous blocks of text out of the context of a larger sequence. It remains to be seen whether ideas that have proven useful at the token level can be applied at this level, too.

[2] Of course, there exist very simple heuristics for finding the header. This discussion is meant to serve as an example of a pattern recognition problem that has many more subtle instantiations.

8.3.5 Information Extraction as Navigation

In order to make good use of layout, perhaps it will be necessary to change the basic problem representation. The representation I adopted assumes that information extraction can be reduced to accepting or rejecting candidate fragments. This representation casts the problem as *classification*, and allows us to apply more or less standard classification techniques, but it is easy to criticize. In particular, it discards context almost completely and pretends that these fragments are so many marbles drawn from an urn, when in fact they overlap densely to form a document. If one fragment is determined to be *almost* a seminar speaker—perhaps, for example, it has one token too many on the right—there is no way to use this information in searching for overlapping fragments.

The *Gedankenexperiment* with the layout of the seminar announcement suggests an alternative problem representation. Rather than classification, can we profitably treat information extraction as *navigation*? Clearly, there is a strong navigational component in a human's approach to the task. What would it take to emulate this algorithmically? Perhaps we can imagine something like a robotic window that starts its search at the upper left-hand corner of a document and is charged with the task of wandering around in search of the seminar speaker. We could equip this robot with a number of sensors, some of them simple derivatives of the kinds of features used in this thesis, others built to measure things like global characteristics of layout.

Results reported at MUC, as well as those presented in this thesis, suggest that there is plenty of room for improvement on the problem of information extraction. Perhaps we have reached a ceiling of sorts, and maybe the best hope for further progress involves emulating the human approach to the task. If this entails actually understanding the contents of a document, then we will have to wait until natural language understanding matures. A premise of information extraction as a field of study, however, is that useful things can be gleaned from a document *without* understanding it. This dissertation adds to the evidence that this is true. It remains to be seen what additional approaches are currently within our reach.

Appendix A

Domains

For this thesis I have experimented with several information extraction domains. Here, I describe the three domains that form the basis of the reported results.

A.1 Seminar Announcements

The seminar announcement collection consists of 485 postings to electronic bulletin boards that are part of the online environment of the Computer Science Department at Carnegie Mellon University. The purpose of every document in this collection is to announce an upcoming project meeting or seminar. As motivation for this domain, we imagine an intelligent agent that can monitor university bulletin boards and summarize upcoming seminars for its user. Such an agent might also attempt to infer whether a seminar is interesting, and might insert details of the seminar, when ordered to do so, into the appropriate slot of the user's electronic calendar. In our collection, these details are presented in a variety of ways (see Appendix B). More often than not, they do not occur in full, grammatical sentences.

We defined four fields for these experiments:

speaker The name of the seminar speaker, including honorifics. First names by themselves are not considered instances, but surnames are, when preceded by honorifics.

location The location of the seminar, typically the name or number of a room.

stime The time at which the seminar is scheduled to begin.

etime The time at which the seminar is scheduled to end.

Of these four fields, the speaker field is the most difficult, start time easiest. Whereas start time tends to be listed in relatively regular contexts, the speaker often appeared as part of a prosaic description of the event. I defined both start and end times to emphasize the need for disambiguation from context.

Some characteristics of the document collection are shown in Table A.1, and of the individual fields in Table A.2. "Strawman accuracy" is the performance of an algorithm that issues random guesses in the manner described in Chapter 2.

Number of files:	485
Smallest:	99 tokens
Largest:	3175 tokens
Mean size:	335 tokens

TABLE A.1: Corpus statistics for the seminar announcement domain.

	speaker	*location*	*stime*	*etime*
Number in corpus	757	643	982	433
Distinct phrases	491	243	151	93
Number of files with	409	464	485	228
Minimum in file	0	0	1	0
Maximum in file	9	4	4	3
Mean number in file	1.6	1.3	2.0	0.9
Smallest (in tokens)	1	1	1	1
Largest (in tokens)	11	14	7	7
Mean size (in tokens)	2.7	3.8	3.6	3.7
Strawman accuracy	1.2%	0.7%	1.4%	1.0%

TABLE A.2: Field statistics for the seminar announcement domain.

A.2 Newswire Articles on Acquisitions

The acquisitions domain contains 600 articles on corporate acquisitions taken from the Reuters data set (Lewis, 1992). Reuters, a standard source of data for experiments in document classification, consists of 21,578 newswire articles produced by the Reuters press service in 1987. Along with its text and title, each article in the collection has been manually assigned zero or more class labels, which are intended to represent the subject of the article.

Among the most populous of the Reuters classes is the "acquisition" category, consisting of 2253 articles. The theme of a typical article in this class is the buying or selling of some corporate entity or asset by another corporate entity or person. In the majority of cases, this transaction is the focus of the article; in others, the article describes some event associated with continuing negotiations. Article detail varies widely, from one-sentence mentions of an acquisition in the offing, to multi-paragraph descriptions of the parties involved.

In designing an information extraction domain from this dataset, I imagined a business user for whom it is important to know the most salient facts of the case—e.g., who is buying whom, and for how much. In deciding what kinds of information would be useful to

acquired	Entity that is purchased
purchaser	Purchasing company or person
seller	Selling company
acqabr	Short name for *acquired*
purchabr	Short name for *purchaser*
sellerabr	Short name for *seller*
acqloc	Location of *acquired*
acqbus	Description of *acquired*'s business
dlramt	Purchasing price
status	Status of negotiations

FIGURE A.1: Fields defined for the acquisitions domain.

this person, I took my cues from the articles themselves, defining fields which were instantiated with reasonable frequency in the articles I reviewed. In addition to the buyer, seller, and purchased company or asset, articles typically reported the status of negotiations—beginning, tentatively completed, completed, broken off—and how much was paid. Also, it is common to provide a little background information about the purchased company, such as location and line of business.

Figure A.1 shows the ten fields I converged on. Note that, in addition to the kinds of information described in the preceding paragraph, I defined three fields for the short names of companies typically used in the body of the article, after a paragraph listing the parties in terms of their official names. I assumed that a user of the system would want the official names, but I did not want to discard the potentially valuable information present in these short references. I imagined that these short names would be easier for a learning system to identify, because much more frequent, and that, having identified them, the system could improve its performance on the official names. Thus, the short names were defined less with a human user in mind, than as a partial decomposition of the domain.

Perhaps it does not need to be noted that not all articles fit readily into this structure. First, there was a considerable number that did not correspond to this pattern at all, and which I passed over in defining the domain. These included a few articles, which occasionally occur in the general Reuters dataset, that consist only of a headline. There were also long articles which either did not focus on a single acquisition—an article, for example, describing the reaction of the U.S. Congress to the spate of acquisitions of American companies by foreign investors—or had as their subject some narrow aspect of an acquisition

Number of files:	600
Smallest:	37 tokens
Largest:	811 tokens
Mean size:	146 tokens

TABLE A.3: Corpus statistics for the acquisitions domain.

event, the details of which were assumed to be known, so were not given in the article. I excluded these, as well.

Among the articles I did include, however, there was wide variance in the degree to which the extraction template could be conveniently filled out. Perhaps the majority of articles fit the mold without much ambiguity, but a sizeable number were close enough in character to the excluded articles described above to make the decision whether to include them difficult. In addition, corporate acquisitions can be considerably more complicated than the flat template structure I adopted suggests. In some cases, they involve outright purchase. In others, they involve simply an increase in holdings of a companies public shares, from, say, 25% to 38%. In still other cases, they involve something difficult to phrase concisely, such as changing the rights and privileges of the board members of a child company in exchange for increased say in company direction. Enriching the template structure could make it better suited to the task of summarization at the expense of increased difficulty in labeling, as well as, I believe, in learning. In an information extraction application, one would want to put considerable thought into this "adequacy-tractability" trade-off. Since my focus is the learning methods that I expect to perform the extraction, I opted for the simplest, most transparent representation.

The fields, too, varied in ease of instantiation. All articles included the full names of the buyer, seller, and bought company, when they existed. Also, the short name fields were, by definition, easy to identify, when present. Text corresponding to the other fields, however, sometimes violated the assumption that field instantiations will be short and contiguous. For example, the terms under which an acquisition is completed sometimes take several sentences to explain; consequently, I only instantiated a *dlramt* field when the article presented a succinct monetary amount as the value of the deal. No real entity described in the articles corresponded to the *status* field; rather, I took short, suggestive noun or verb phrases, such as "completed" or "agreement in principle," which I assumed would convey the state of negotiations to a human reader. Finally, *acqloc* and *acqbus* shared a particular characteristic: Although it was easy to identify these fields in text, they sometimes allowed single-token instantiations (e.g, "bank") and other times took up the better part of a sentence (e.g., "Oregon, Washington, and several cities in California").

Table A.3 shows some characteristics of the document collection. Table A.4 summarizes characteristics of individual fields.

	acquired	*purchaser*	*seller*	*acqabr*	*purchabr*	*sellerabr*	*acqloc*	*acqbus*	*dlramt*	*status*
Number in corpus	683	624	267	1450	1263	431	213	264	283	461
Distinct phrases	589	533	232	609	627	256	168	236	181	189
Number files with	593	545	235	437	445	182	178	230	259	453
Minimum in file	0	0	0	0	0	0	0	0	0	0
Maximum in file	8	11	5	24	18	12	7	4	3	2
Mean in file	1.1	1.0	0.4	2.4	2.1	0.7	0.4	0.4	0.5	0.8
Smallest (tokens)	1	1	1	1	1	1	1	1	1	1
Largest (tokens)	13	17	10	6	8	6	14	13	9	12
Mean size	3.5	3.2	3.2	1.4	1.4	1.5	2.8	3.3	3.2	2.2
Strawman accuracy	1.4%	1.3%	1.3%	7.0%	5.8%	4.6%	1.4%	1.0%	1.2%	0.9%

TABLE A.4: Field statistics for the seminar announcement domain.

A.3 University Web Pages

The "University Web Pages" domain is really a set of related domains defined for samples from the same large document collection. This collection was constructed as part of the World Wide Knowledge Base (WebKB) effort, the goal of which is a system capable of automatically constructing knowledge bases by browsing the Web. The focus of the project is the use of machine learning methods to support this effort. Our experiments in these domains constitute one component of a multi-pronged initiative, which includes explorations in statistic document classification, and classification of pages based on Web connectivity, among other things.

The data set from which I sampled to create these domains consists of 4,127 Web pages associated with computer science departments from four large university: Cornell, University of Texas, University of Washington, and University of Wisconsin. As part of initial experiments, these pages were labeled by hand to indicate membership in seven mutually exclusive categories: *department*, *faculty*, *staff*, *student*, *research project*, *course*, and *other*.

A.3.1 Course Pages

The Course Pages domain is a sample of pages labeled *course* in the WebKB data set, and some additional pages I collected while labeling. Only "top-level" course pages receive the *course* label in this collection; associated pages, such as those listing assignment due dates, are labeled *other*. I found that this labeling scheme missed some pages containing instances of fields I wanted to define, so I treated the WebKB labeling as a guide and browsed around the pages labeled *course* to find replacements, more current pages, or supplementary pages.

For this domain I defined three fields:

crsNumber The university code identifying the course

crsTitle The official title of the course

crsInst The names of all instructors and teaching assistants

Number of files:	101
Smallest:	89 tokens
Largest:	2631 tokens
Mean size:	633 tokens

TABLE A.5: Corpus statistics for the WebKB course pages sub-domain.

	crsNumber	*crsTitle*	*crsInst*
Number in corpus	296	165	216
Distinct phrases	86	70	149
Number of files with	94	88	95
Minimum in file	0	0	0
Maximum in file	10	12	25
Mean number in file	2.9	1.6	2.1
Smallest (in tokens)	1	1	2
Largest (in tokens)	4	7	6
Mean size (in tokens)	1.8	3.2	2.3
Strawman accuracy hline	3.3%	1.4%	0.9%

TABLE A.6: Field statistics for the WebKB course pages sub-domain.

The *crsInst* field is one of two "many-per-document" (MPD) fields used in thesis experiments. Instances of all fields were manually annotated.

Table A.5 presents statistics of the WebKB course pages corpus. Table A.6 lists characteristics of the individual fields.

Number of files:	96
Smallest:	26 tokens
Largest:	2242 tokens
Mean size:	421 tokens

TABLE A.7: Corpus statistics for the WebKB project pages sub-domain.

	projTitle	*projMember*
Number in corpus	352	747
Distinct phrases	73	578
Number of files with	80	66
Minimum in file	0	0
Maximum in file	28	39
Mean number in file	3.7	7.8
Smallest (in tokens)	1	2
Largest (in tokens)	10	8
Mean size (in tokens)	2.2	2.3
Strawman accuracy	7.8%	7.2%

TABLE A.8: Field statistics for the WebKB project pages sub-domain.

A.3.2 Research Project Pages

The Research Project Pages domain was created in a way identical to the Course Pages domain. For this domain I defined two fields:

projTitle The title given to the project by its members

projMember The names of all members, including principle investigators, graduate and undergraduate members, affiliated researchers, and project alumni

The resulting collection consisted of 96 annotated project pages. The *projMember* field is the second of two MPD fields used in thesis experiments. Tables A.7 and A.8 present corpus and field statistics, respectively, for this domain.

Appendix B

Excerpts

This appendix presents excerpts of documents from the three thesis domains. For the reader with no experience in information extraction, the discussion in Chapter 2 will help provide a basic understanding but may leave many holes in his or her intuition for the problem. Similarly, without experience in any of the domains discussed, researchers in information extraction and related disciplines may have difficulty relating the work described in this thesis to their own. This appendix is an attempt to compensate for these deficits.

Excerpts are presented with as much context as possible—entire short documents in many cases. My object is to illustrate the range, and not necessarily to reflect the distribution, of styles and subject matter in each domain. Both common and some interesting uncommon patterns are included. $\boxed{\text{Boxes}}$ identify instances of fields that are the targets of thesis experiments.

B.1 Seminar Announcements

The seminar announcements differ from the other two domains in the preponderance of their use of simple labels to communicate essential details. Often, important details of upcoming seminars, including those that are targets of information extraction, are preceded by a suggestive phrase and a colon. Table B.1 presents an excerpt that is unusual, in that *all* essential details are presented in this way. The *label/colon* device is used for one or more of the four seminar announcement fields in a majority of documents. In few of the documents, however, is it used as extensively as in the figure.

Many seminar announcements convey essential details in a single short paragraph consisting of one or a few sentences. Most such paragraphs are reasonably well-formed grammatically, although they often contain terms unlikely to appear in the lexicon of a general-purpose NLP system, terms such as “nano-rheology” the names of speakers. The paragraph in Table B.2 is typical of the form of such announcements, but atypical in its ungrammaticality. Note that here, too, the label/colon device is used to announce the start time, which is listed again (in a slightly different form) in the message body. Both *stime* instances are

```
<0.26.4.95.11.09.31.hf08+@andrew.cmu.edu.0>
Type:     cmu.andrew.academic.bio
Topic:    "MHC Class II: A Target for Specific Immunomodulation of the
          Immune Response"
Dates:    3-May-95
Time:     3:30 PM
Place:    Mellon Institute Conference Room
PostedBy: Helena R. Frey on 26-Apr-95 at 11:09 from andrew.cmu.edu
Abstract:


 Seminar: Departments of Biological Sciences
 Carnegie Mellon and University of Pittsburgh
 Name: Dr. Jeffrey D. Hermes
 Affiliation: Department of Autoimmune Diseases Research & Biophysical Chemistry
 Merck Research Laboratories
 Title: "MHC Class II: A Target for Specific Immunomodulation of the
 Immune Response"
 Host/e-mail: Robert Murphy, murphy@a.cfr.cmu.edu
 Date: Wednesday, May 3, 1995
 Time: 3:30 p.m.
 Place: Mellon Institute Conference Room
 Sponsor:  MERCK RESEARCH LABORATORIES

Schedule for 1995 follows: (as of 4/26/95)
Biological Sciences      Seminars          1994-1995
Date        Speaker        Host
April 26        Helen Salz        Javier L˜pez
May 3        Jefferey Hermes        Bob Murphy
MERCK RESEARCH LABORATORIES
```

TABLE B.1: A complete seminar announcement illustrating the common use of the label/colon device.

```
<0.22.2.95.09.47.47.ed47+@andrew.cmu.edu.0>
Type:     cmu.andrew.official.cmu-news
Topic:    Physics Colloquium,Feb. 27
Dates:    27-Feb-95
Time:     4:30 PM
PostedBy: Edmund J. Delaney on 22-Feb-95 at 09:47 from andrew.cmu.edu
Abstract:

Physic Colloquium, Monday, Feb. 27,  Steve Granick, University of
Illinois, Urbana, "Soft matter in a tight spot:  nano-rheology of
polymers and complex fluids," 4:30 p.m., 7500 Wean Hall, Coffee at 4:15
p.m.
```

TABLE B.2: A complete seminar announcement illustrating the use of a single short paragraph to convey essential details.

```
<0.23.5.95.16.22.55.ed47+@andrew.cmu.edu.0>
Type:     cmu.andrew.official.cmu-news
Topic:    Psychology Post-Doc Talk
Dates:    25-May-95
Time:     12:00 PM
PostedBy: Edmund J. Delaney on 23-May-95 at 16:22 from andrew.cmu.edu
Abstract:

         Patricia Brooks
                      Post-doctoral assistant
                      Department of Psychology
                     Carnegie Mellon University

    Phonological and Semantic Priming in Children's Picture Naming

                      Thursday May 25, 1995
                           Baker Hall 355
                              12 noon
Phonological and semantic priming was explored in children using a
cross-modal picture-word interference paradigm.  Pictures of familiar
objects (e.g., snake, moon, hand) were presented to 5- to 11-year-olds
and adults on a computer screen while digitized auditory stimuli (words
and non-words) were presented simultaneously over head-phones. The
auditory stimuli were related (phonologically or semantically) or
unrelated to the names of the pictures. Children were instructed to name
the pictures as quickly as possible while ignoring the auditorily
presented items. Children were tested over multiple sessions to vary the
stimulus onset asyncrony (SOA--the interval between presentation of the
prime and presentation of the picture-to-be-named). Robust phonological
priming occurred which was influenced by the lexical status of the prime
(whether it was a non-word or a real word) and by whether the prime
shared the onset consonant or rhymed with the name of the target
picture.  In contrast, semantically-related auditory stimuli did not
facilitate children's picture naming.
```

TABLE B.3: A complete seminar announcement illustrating the mixing of prose with other devices and the use of centering.

counted for the purposes of training, and extraction of either counts as a correct response during testing.

Most of the seminar announcements mix prose paragraphs and a variety of other presentational devices. One particularly common device is centering, illustrated in Table B.3. Essential details are approximately centered (centering of the speaker's name in Table B.3 is very approximate!) at the top of the message body without labels or other text. Full paragraphs of grammatical text follow—often the talk abstract or biographical information on the speaker. Of course, though common, centering is not the only layout device used. In Table B.4, the author appears to be emulating features of typeset text, such as itemization, italics and headlines. Note that this announcement does not list a speaker, although it does include a name.

Table B.5 illustrates the use of a device that is uncommon in this domain, a table. The approach to learning taken in this thesis, the representation of the text as a sequence of

```
<0.28.4.95.10.13.54.rf51+@andrew.cmu.edu.1>
Type:     cmu.andrew.assocs.gso
Topic:    UTC Summer Seminars for Graduate Students
Dates:    24-May-95
Time:     2:00 - 4:00 PM
PostedBy: Rea Freeland on 28-Apr-95 at 10:13 from andrew.cmu.edu
Abstract:

University Teaching Center
Support for Graduate Students -- Summer 1995

The combination of activities below is designed to assist current and
prospective TAs as well as graduate students who are preparing for
future academic positions involving teaching.  The seminars were
selected to be repeated based on attendance in and feedback about
previous series.  Those of you interested in other specific issues may
be interested in programs we
conduct in addition to the seminars.

Included below is information on the following programs:
- Summer Seminars on Teaching
- Videotaping and Feedback Opportunities
- Observations and Consultations
- Ongoing Reading/Discussion Group
- Documentation of Teaching Development

***Pre-registration is required; please see the end of this post for
instructions. ***

SEMINARS ON TEACHING
The six seminars below count toward the Documentation of Teaching
Development Program.

Overview of Student Motivation
        Wednesday, May 24, 2:00 - 4:00 PM, Carnegie Conference Room, Warner Hall
Instructors often take for granted that our students have the motivation
to learn what we teach,  but the level or type of motivation varies
greatly with different courses, assignments and students.  This seminar
will provide a framework of theories of motivation and provide
opportunities to discuss how instructors can engage students more fully
in learning in their courses.
```

TABLE B.4: A complete seminar announcement illustrating how itemizations, italics, and headlines are emulated.

```
<0.2.10.94.14.09.41.dd7a+@andrew.cmu.edu.9>
Type:     cmu.andrew.org.epp
Topic:    CEE EES Seminar Series
Dates:    21-Oct-94
Time:     [2:30]
PostedBy: David Adam Dzombak on 2-Oct-94 at 14:09 from andrew.cmu.edu
Abstract:
                DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING
                          CARNEGIE MELLON UNIVERSITY

                 ENVIRONMENTAL ENGINEERING AND SCIENCE SEMINARS
                                  FALL 1994

DATE      ROOM      TIME   SPEAKER               TOPIC

10/21c  [PH A18C]  [2:30] [Mr. Rajat Ghosh]  Modeling of In Situ
                             CEE             Solvent Extraction of
                                             Coal Tar: Interrupted
                                             Pumping Process
```

TABLE B.5: A complete seminar announcement illustrating the use of an ad hoc table.

```
{[MERIDIAN ENERGY]}, [CASTONE] END LETTER OF INTENT
    CHICAGO, March 26 - {[Meridian Energy Inc]} and [Castone
Development Corp], a privately-held company, jointly announced
that they have decided to [terminate the letter of intent] under
which [Meridian] would have acquired [Castone].
 Reuter
```

TABLE B.6: A typical short article from the acquisitions domain.

tokens, cannot cope with such structures. While an understanding of layout, of the text as a two-dimensional object, appears useful in some of the other excerpts, here it is critical. Note that the talk title in this fragment cannot be extracted by any system under the linear representation. This suggests that a domain-independent solution to information extraction cannot ignore layout.

B.2 Acquisitions Articles

The format of acquisitions articles varies very little from document to document. Each such document consists of a headline, a date line, and one or more paragraphs of journalistic prose. Rather, articles vary stylistically and in their subject matter. They may adhere to journalistic conventions that appear to govern the narration of acquisitions, or they may follow an unconventional style. They may describe an acquisition directly, or they may

```
{VIA} SETS RECORD DATE FOR MERGER VOTE
    NEW YORK, March 26 - Viacom International Inc said it set
April 6 as the record date for shareholders entitled to vote at
a special meeting to be held to vote on the proposed merger of
Arsenal Acquiring Corp, a wholly-owned subsidiary {Arsenal
Holdings Inc} into Viacom.
    It said the date of the special meeting has not yet been
determined.
 Reuter
```

TABLE B.7: A complete acquisitions article, the subject of which is not directly a proposed or completed acquisition but a byproduct of one.

```
VIDEO DISPLAY {VIDE} TO SELL CABLE TV UNIT
    ATLANTA, March 3 - Video Display Corfp said it has reached
a tentiative agreement to sell its existing cable television
business for undisclosed terms and expects to report a gain on
the transaction.  The buyer was not named.
    The company said it will redeploy its service assets into
manufacturing and distribution.
    It said the operations being sold accounted for about five
pct of revenues for the year ended February 28 and lost money.
 Reuter
```

TABLE B.8: A complete acquisitions article in which many details, including the buyer, are not listed.

focus on an event only associated with an acquisition.

Tables B.6 and B.7 present two articles that differ in this second way. The article in Table B.6 directly describes the outcome of an acquisition negotiation. The article in Table B.7 relates an event associated with such a negotiation—a shareholder's meeting to vote on the proposed acquisition. This distinction may be of relevance for information extraction. The language of the former kind of article tends to follow very regular conventions. Typically a company "says" or "announces" that something has happened, an acquisition has been completed, for example, or a rival has made a bid for outstanding shares of the company's stocks. In the latter kind of article, a company may make an announcement, but about something ancillary to the acquisition process. This may result in difficulty for the human labeler to fit the article into the template defined for the domain. In Table B.7 I tagged the phrase "record date" as an instance of the *status* field, but this does not fit the semantics of this field well, as defined by other instances in the corpus. As a consequence, it is unlikely that any learner would correctly extract this phrase.

```
ALCAN AUSTRALIA BIDS FOR ALCAN NEW ZEALAND
    SYDNEY, March 27 - Alcan Australia Ltd {AL.S} said it will
make a 39.3 mln N.Z. Dlr cash bid for all the issued shares of
Alcan New Zealand Ltd at 1.80 N.Z. Dlrs each with a
four-for-three share alternative.
    Both are 70 pct owned by Canada's Alcan Aluminium Ltd AL
which will take the share swap option, Alcan Australia deputy
chairman Jeremy Davis said in a statement. The remainder of
Alcan New Zealand's totalled issued 21.84 mln shares are
broadly held while Alcan Australia's are primarily held by
institutions. Alcan NZ last traded at 1.55 NZ dlrs, while Alcan
Australia today ended four cents down at 1.15 dlrs.
    Davis said the offer, which is subject to approval by the
New Zealand Overseas Investment Commission, was a response to
the integration of the two countries' markets under the
Australia-New Zealand Closer Economic Relations treaty.
    Alcan New Zealand shareholders who accept the offer would
also receive the final dividend of 10 cents a share normally
payable on May 27.
    Alcan Australia would invite New Zealand representation to
its board and would apply to list its shares on the New Zealand
Stock exchange, Davis said.
 REUTER
```

TABLE B.9: A complete acquisitions article illustrating some of the subtleties involved in distinguishing the roles of companies.

```
GERBER {GEB} SETS DEADLINE FOR UNIT'S BUYOUT
    FREMONT, MICH., April 3 - Gerber Products Co said it has
given management of its CWT Inc trucking subsidiary 60 days to
pursue a leveraged buyout of the subsidiary.
    It said CWT Inc, which has operations in the Midwest and
Southeast, has annual revenues of approximately 135 mln dlrs.
 Reuter
```

TABLE B.10: Another acquisitions article in which the parties are identifiable but difficult to extract.

Acquisitions articles also vary in the amount of detail they provide. Table B.8 adheres to conventions of presentation common in this domain, but many of the usual details are missing. Also, instead of a company name, the *acquired* field is instantiated here as "cable television business." While it is more common for company names to appear in this field, occasionally an asset or resource, as in this article, is the object of purchase.

Tables B.9 and B.10 present two articles that illustrate how some of the assumptions that underlie the definitions of fields are stretched by some of the articles. The canonical

```
INVESTOR GROUP PUTS PRESSURE ON GENCORP {GY}
    By Patti Domm, Reuters
    NEW YORK, March 30 - An investor partnership, seeking to
acquire GenCorp Inc, said it would attempt to unseat the
company's board of directors and take other hostile actions if
the firm refuses to discuss its 2.3 billion dlr takeover bid.
    General Acquisition Co, comprising investors Wagner and
Brown and glass-maker AFG Industries, also reiterated its
willingness to negotiate with Gencorp.
    The partnership has earlier offered 100 dlrs per share for
GenCorp -- a tire, broadcasting, plastics and aerospace
conglommerate.
...
```

TABLE B.11: Beginning of an acquisitions article in which one of the main parties is not mentioned in the first paragraph.

acquisition event involves a company (the *purchaser*) buying another company (the *acquired* company), possibly from a third company (the *seller*). It is more or less difficult to fit each of these articles into this mold. In the article in Table B.9 one company makes a bid for its sibling. The names of the siblings, as well as that of the parent company, which is also listed in the article, are all similar to each other, enough so that an automatic system might easily become confused. Note that "Alcan New Zealand" occurs lower in the article but is not tagged. Mis-tagging or failure to tag instances is noise which may or may not affect learner performance, but which is an almost inevitable feature of any hand-labeled extraction problem.

The relationships between the parties are even stranger in the article in Table B.10. The management of a company is seeking to purchase the company itself from the parent company. It is difficult to tag such an article in a satisfactory way. As it is, the *acquired* field is instantiated simply as "management." In this article we also see typical examples of how *acqbus* ("trucking") and *acqloc* ("Midwest and Southeast") are instantiated.

The article in Table B.11 deviates considerably from conventions of presentation. As is very often true, the main verb of the first sentence is "said," but its subject, the *purchaser*, is an anonymous "investor partnership." The correct instantiation of *purchaser*, however, does not occur until the beginning of the second paragraph—"General Acquisition Co." If a system is to make the correct extraction in this case, it evidently needs to associate phrases such as "an investor partnership" and "the partnership" with the actual instance, which does not occur in the same sentence as either of these phrases. In other words, it needs to perform anaphora resolution. Note, incidentally, the instantiation of *acqbus*—"tire, broadcasting, plastics and aerospace"—which is quite different from, and longer than, the instantiation in Table B.10.

Purchasers and sellers are occasionally people, as in Table B.12. Because we are not guaranteed that instances of *acquired*, *purchaser*, and *seller* are all companies, named entity

```
INVESTOR HAS 8.0 PCT OF ALLEGHENY INT'L {AG}
    WASHINGTON, March 30 - A group of firms and funds
controlled by New York investor Mario Gabelli said it has
acquired the equivalent of 882,507 shares of Allegheny
International, or 8.0 pct of the total outstanding.
    In a filing with the Securities and Exchange Commission,
the Gabelli group said it bought the stake as part of its
business and not in an effort to seek control of the company.
    It said it may by more shares or sell some or all of its
current stake. The stake includes 782,000 common shares and
cumulative convertible preferred stock which could be converted
into 100,507 common shares.
 Reuter
```

TABLE B.12: A complete acquisitions article in which one of the main parties is an individual, rather than a company.

recognition is less useful as a pre-processing step than it might otherwise be. Named entity recognition is the problem of identifying instances of generic classes such as people and companies without determining their specific role in a document. As a pre-processing step for *purchaser*, it clearly would be useful, but at least two such tasks—one for the names of companies, the other for people—would be necessary, meaning additional subsequent work at disambiguation.

B.3 University Web Pages

Table B.13 illustrates a typical form course pages take. The *crsNumber* and/or *crsTitle* fields are instantiated in the HTML title and between <h?>-tags in the body of the page. When the two fields are instantiated together, *crsTitle* often follows *crsNumber*. This excerpt also contains an instance of *crsInst*. Note the paucity of clues surrounding this instance. The fact that this instance is centered was not available to SRV in the experiments presented in Chapter 5, because instead of a <center>-tag, the author used an attribute of the <p>-tag; making such attributes available to SRV in the form of token features might be an interesting direction for future research.

Recall that *crsInst* is a "many-per-document" (MPD) field, i.e., multiple instances can appear in a page, each representing a distinct entity. Tables B.14 and B.15 show two ways in which these instances can occur in course pages. In Table B.14 two instances occur in sequence, each preceded by a suggestive label. In Table B.15, in contrast, they are arranged in a table, and the header of the column in which they occur is the strongest evidence that they are field instances. In experiments with SRV on this domain, I introduced HTML-specific relational features like table_col_header, which maps a token in a column to the first token in the column header (e.g., "Shih" to "TA"). My hope was that this would enable SRV to find patterns such as this one. I saw little evidence, however, that such features

```
<html>
<head>
<title> CS113 Home Page</title>
</head>
<body>

<hr>
<h1>
<p align=center>
CS113:  C Programming
</h1>

<h2>
<p align=center>
David Walker
</h2>
<h3>
<p align=center>
<Fall 1997, weeks 5-8>
</h3>
<hr>
...
```

TABLE B.13: The top of a typical course page from the WebKB domain.

```
...
<H2> <Strong> Professor: Lorenzo Alvisi </Strong> </H2>
Office: Taylor Hall 4.122 <BR>
Phone: 471-9792 <BR>
Email: <A href="..."> lorenzo@cs.utexas.edu
</A> <BR>
Office Hours: Thursday 11:15-12:15. <BR>
Help Sessions: 8:00-9:00AM Monday
and Friday, in Welch 3.402.

<H2> <Strong> Teaching Assistant: Rupert Tang Lap Poon  </Strong> </H2>
Office: UA9 4.108e <BR>
Phone: 471-9755 <BR>
Email: <A href="..."> rupert@cs.utexas.edu
</A> <BR>
Help Sessions: 7:15-8:15PM Thursday and Friday, in Garison 301 <br>
(one more help session to be decided)<br></H4>
...
```

TABLE B.14: A course page excerpt in which instances of *crsInst* are listed linearly.

```
...
<h2><img src="..."> Teaching Assistants:</h2>
<ul>
<table border=2 cellspacing=1>
<tr bgcolor="#CCCCCC">
<td><center><b>TA</td><td><center>e-mail</td></tr>
<tr><td><center> Thomas Yan </td><td><a href="...">tyan@cs.cornell.edu</a></td></tr>
<tr><td><center> Patrick White </td><td><a href="...">white@cs.cornell.edu</a></td></tr>
<tr><td><center> Linda Shih-Chien Lee </td><td><a href="...">sclee@cs.cornell.edu</a><td></tr>
</table>
<br>
<img src="...">
<font color="#C00000">Office hours are now available</font>
<a href="offhours.html"> here</a>.
...
```

TABLE B.15: A course page excerpt in which a HTML table is used to present instances of *crsInst*.

were used effectively.

The *projMember* is also a MPD field and usually is instantiated more frequently in a page than *crsInst*. Table B.16 illustrates a common pattern of instantiation: a series of itemizations, the name of the project member beginning each item, perhaps followed by additional information such as title or email address. This pattern is so common that a wrapper approach seems appropriate, one which can take advantage of embedded repeating structures.

Finally, Tables B.17 and B.18 present two typical contexts for instances of *projTitle*. Like *crsTitle*, instances of *projTitle* are often found in the HTML title or between `<h?>`-tags. Unlike *crsTitle*, *projTitle* instances are also often found in prose in the body of the page, as in Table B.18. Whereas course pages typically serve as an organizational tool for university courses, project pages have the additional function of advertising a research project to the world at large. Consequently, the goals and technical achievements of a project are often described in paragraphs of technical prose. In my experiments I did not attempt linguistic processing of these descriptions, but this might prove very useful for this field.

```
...
<H2><A Name="...">Principle Investigator</A></H2>
<menu>
<LI> <A HREF="...">
  Bart Miller </A> (Professor/Principle Investigator)
</menu>
<BR>

<H2><A Name="...">Research Staff</A></H2>
<menu>
<LI> <A HREF="...">
  Oscar Naim </A> (Assistant Scientist)
<LI> <A HREF="...">
   Brian Wylie </A> (Associate Researcher)
</menu>
<BR>

<H2><A Name="...">Graduate Students</A></H2>
<menu>
<LI> <A HREF="...">
  Matt Cheyney </A> (Research Assistant)
<LI> <A HREF="...">
  Karen Karavanic  (Research  Assistant)
<LI> <A HREF="...">
  Tia Newhall </A> (Research Assistant)
...
```

TABLE B.16: Project page excerpt showing a typical listing of members.

```
<html>
<head>
<title>  Wisconsin Wind Tunnel Project  Home Page </title>
</head>
<body
background="wwt_logo_background.gif" body text="#000000" link="#000fff"
vlink="#000aaa"
>
<hr>
<h1>
<!--  Changed SRC="wwt_logo.gif" so ftp could be used to log accesses -->
<IMG SRC="..." ALT="WWT Logo" ALIGN=MIDDLE>
Wisconsin Wind Tunnel Project
</h1>
<hr>
<p>
...
```

TABLE B.17: Top of a project page showing instances of *projTitle*.

```
<TITLE>Condor Project Homepage</TITLE>
<IMG ALIGN=MIDLLE SRC="...">

<HR>

<H3>
<A NAME="...">Objective:</A>
</H3>

The goal of the Condor project is to develop, implement, deploy, and evaluate
mechanisms and policies that support High Throughput Computing (HTC) on large collections
...
```

TABLE B.18: Top of a project page illustrating instances of *projTitle* in various contexts.

Appendix C

The Tokenizing Library

All of the learners presented in this thesis share a single code substrate, a C library called `libtoken` that takes care of converting documents into the representation presented in Chapter 2: A document is regarded as a sequence of tokens, and instances of a field take the form of unbroken subsequences. This library is compiled into SRV, which is also written in C. The other learners, which are implemented in Perl, make use of a Perl module called `Token` that defines Perl versions of most of the functions defined in `libtoken`.

Any document representation makes some of the information contained in a document immediately accessible, makes some information available by reconstruction, and completely discards some information. Because the set of problems a learner is able to address depends in part on which information is available, it is important to know the details of the representation. This Appendix presents those details. Of particular interest are some subtle differences between `libtoken`'s processing of HTML and that apparently used by related work.

The primary function of `libtoken` is to represent a document as a sequence of tokens. This is accomplished by calling the function

```
void p_parse(char *fname);
```

providing it with the name of the file to process. This function is called once per document. A number of other functions then allow access to the information contained in the document. Calls to these secondary functions implicitly refer to the last document processed by `p_parse`. The document processed by `p_parse` is stored as a single array of tokens. A token is any sequence of characters matching the pattern `[A-Za-z0-9]+` or the pattern `[^A-Za-z0-9 \t\n]`, i.e., either an unbroken sequence of alphanumeric characters or a single non-whitespace character. Note that `libtoken` also stores information about whitespace, but it is neither represented in the central array nor easily accessible to client code.

An index in the global array is assigned to each token matching the above patterns. Once parsing is complete, client code can access information about the contents of a document by means of functions like

```
Time: <stime>3:30 p.m</stime>.
Place: <location>Mellon Institute Conference
Room</location>
```

TABLE C.1: Excerpt from a seminar announcement showing annotation used to identify field instances.

```
unsigned p_token_count(void);
```

which returns the number of tokens in a document;

```
char *p_token(unsigned index, char buf[], unsigned bufsz);
```

which returns a copy of the token occurring at position `index` in the document; and

```
char *p_fragment(unsigned from, unsigned to, char buf[], unsigned bufsz);
```

which returns a copy of the text beginning with the token at index `from` and extending to but not including the token at index `to`, including any whitespace.

As part of labeling for information extraction, field instances are identified directly in documents by means of SGML-style tags. Table C.1 shows an excerpt from a seminar announcement, complete with the tags that mark instances of two fields. One of the jobs of `libtoken` in tokenizing a document is to remove these tags while storing information concerning their position. In other words, these tags are not stored as tokens in the global array, and the functions described in the previous paragraph act as if these tags did not appear. This ensures that the essential information in a document, including its whitespace patterns, is not altered by the labeling process.

Instead, a number of additional functions are defined which answer questions about the positions of, and information contained in, the tags. The two most important such functions are:

```
int p_in_field(unsigned index, char *field);
```

a Boolean function returning true if the token at `index` is in scope of tag whose name string matches the string in `field`, and

```
int p_fields_at(unsigned index, char buf[], unsigned bufsz, unsigned fbuf[], unsigned fbufsz);
```

which returns an array of all fields in force at position `index`.

Because the tags used to mark field instances are identical in form to HTML tags, `libtoken` "understands" HTML. One consequence of this understanding is that, like annotations introduced for information extraction, HTML tags are removed from the internal document representation, and information about them is only made available through calls to functions like those listed in the previous paragraph. What a client of libtoken sees of a Web page, therefore, is any text left over once tags have been removed.

The effect of this representation when working with HTML can be good or bad, depending on the intended application. For example, because libtoken also stores tag attribute information, which it makes available by additional functions, it is trivial to retrieve the set of URLs referenced in a page, as well as the text associated with each. It is easy to say whether a token occurs within scope of a tag, however large the tag's scope. Determining whether it occurs *next to* a tag, however, requires at least two calls to libtoken's functions.

Rote and BayesIDF are handicapped with respect to HTML because, as a consequence of their reliance on libtoken, HTML tags are invisible to them. In contrast, SRV is provided with a set of HTML features that query libtoken functions. Even for SRV, however, some observations are difficult; in order to assert that a token occurs at the boundary of a HTML field, SRV must make two assertions: that the token occurs in the field *and* that the previous or next token occurs outside it. In contrast, much of the work on information extraction from HTML, including work on wrapper induction, assumes that HTML tags are directly visible to the learner. This may result in more competent extractors. It would be possible to modify libtoken so that tags other than those used for annotation are visible. For the sake of simplicity and clarity, however, I did not take this approach.

References

D. Angluin. Inference of reversible languages. *Journal of the Association for Computing Machinery*, 29(3):741–765, 1982.

C. Aone and S. W. Bennett. Applying machine learning to anaphora resolution. In S. Wermter, E. Riloff, and G. Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pages 302–314. Springer-Verlag, Berlin, 1996.

D. E. Appelt, J. R. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, and M. Tyson. SRI International FASTUS system MUC-6 test results and analysis. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, 1995.

C. Apté, F. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, July 1994.

S. E. August and C. P. Dolan. Hughes Research Laboratories: description of the trainable text skimmer used for MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 189–196, June 1992.

D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, 1997.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Working Papers of ACL-97 Workshop on Natural Language Learning*, 1997.

M. E. Califf. *Relational Learning Techniques for Natural Language Information Extraction*. PhD thesis, University of Texas at Austin, August 1998.

C. Cardie. A case-based approach to knowledge acquisition for domain-specific sentence analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 798–803, July 1993.

C. Cardie. Empirical methods in information extraction. *AI Magazine*, 18(4):65–79, 1997.

R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In R. C. Carrasco and J. Oncina, editors, *Grammatical Inference and Applications: Second International Colloquium, ICGI-94*. Springer-Verlag, September 1994.

N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.

B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, 1990.

P. Chan and S. Stolfo. Experiments on multistrategy learning by meta-learning. In *Proceedings of the Second International Conference on Information and Knowledge Management (CIKM 93)*, pages 314–323, November 1993.

P. Clark and R. Boswell. Rule induction with CN2: some recent improvements. In Y. Kodratoff, editor, *Machine Learning – EWSL-91*, pages 151–163. Springer-Verlag, Berlin, 1991.

W. W. Cohen. Learning to classify English text with ILP methods. In L. D. Raedt, editor, *Advances in Inductive Logic Programming*, IOS Frontiers in AI and Applications. IOS Press, 1995.

J. Cowie, L. Guthrie, W. Jin, R. Wang, T. Wakao, J. Pustejovsky, and S. Waterman. CRL/Brandeis: description of the Diderot system as used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pages 161–179, August 1993.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

Defense Advanced Research Projects Agency. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, McLean, Virginia, June 1992. Morgan Kaufmann Publisher, Inc.

Defense Advanced Research Projects Agency. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Baltimore, Maryland, August 1993. Morgan Kaufmann Publisher, Inc.

Defense Advanced Research Projects Agency. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Morgan Kaufmann Publisher, Inc., November 1995.

P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96)*, pages 105–112, July 1996.

P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24(2):141–168, 1996.

R. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the world-wide web. In *Proceedings of the First International Conference on Autonomous Agents*, 1997.

R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973.

P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference? In *Grammatical Inference and Applications*, volume 862 of *Lecture Notes in Computer Science*, pages 25–37. Springer Verlag, 1994.

D. A. Evans, N. D. Brownlow, W. R. Hersh, and E. M. Cambpell. Automating concept identification in the electronic medical record: An experiment in extracting dosage information. In J. J. Cimino, editor, *Proceedings, 1996 AMIA Annual Fall Symposium*, Journal of the American Medical Informatics Association (JAMIA), Symposium Supplement, pages 388–392, Philadelphia, PA, 1996. Hanley & Belfus, Inc.

D. Freitag. Using grammatical inference to improve precision in information extraction. In *Notes of the ICML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*, 1997. http://www.cs.cmu.edu/p̃dupont/ml97p/ ml97_GI_wkshp.tar.

Y. Freund and R. E. Shapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

T. Goan, N. Benson, and O. Etzioni. A grammar inference algorithm for the World Wide Web. *Working Notes of the AAAI-96 Spring Symposium on Machine Learning in Information Access*, 1996.

E. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

E. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

A. Kehler. Probabilistic coreference in information extraction. In *Proceedings of the Second Conference On Empirical Methods in Natural Language Processing (EMNLP-2)*, 1997.

J.-T. Kim and D. I. Moldovan. Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactions on Knowledge and Data Engineering*, pages 713–724, October 1995.

R. Kohavi. The power of decision tables. In *Proceedings of the European Conference on Machine Learning (ECML-95)*, pages 174–89, April 1995.

N. Kushmerick. *Wrapper Induction for Information Extraction*. PhD thesis, University of Washington, 1997. Tech Report UW-CSE-97-11-04.

W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. University of Massachusetts: description of the CIRCUS system as used for MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, June 1992.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th International Conference on Research and Development in Information Retrieval*, July 1994.

D. Lewis. *Representation and Learning in Information Retrieval*. PhD thesis, Univ. of Massachusetts, 1992. CS Tech. Report 91–93.

D. D. Lewis. Reference list to accompany sigir-97 tutorial on machine learning for information retrieval, 1997. `http://www.research.att.com/ lewis/papers/lewis98.ps`.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

H. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2:159–165, 1958.

M. Maron. Automatic indexing: An experimental inquiry. *Journal of the Association for Computing Machinery*, 8:404–417, 1961.

J. F. McCarthy and W. G. Lehnert. Using decision trees for coreference resolution. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.

R. Michalski and G. Tecuci, editors. *Machine Learning: A Multistrategy Approach*. Morgan Kaufmann, San Mateo, CA, 1994.

R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134. Tioga Publishing Company, Palo Alto, Ca, 1983.

G. Miller. WordNet: A lexical database for English. *Communications of the ACM*, pages 39–41, November 1995.

T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

T. M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., 1997.

K. Muraki, S. Doi, and S. Ando. NEC: description of the VENIEX system as used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pages 147–159, August 1993.

I. Muslea, S. Minton, and C. Knoblock. Wrapper induction for semistructured, Web-based information sources. In *Proceedings of the Conference on Automatic Learning and Discovery (CONALD-98)*, 1998.

W. W. Noah and R. V. Weeks. TRW: description of the DEFT system as used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pages 237–248, August 1993.

J. Oncina and P. Garcia. Inferring regular languages in polynomial update time. In N. P. de la Blanca, A. Sanfeliu, and E. Vidal, editors, *Pattern Recognition and Image Analysis*, volume 1 of *Series in Machine Perception and Artificial Intelligence*, pages 49–61. World Scientific, 1992.

T.-W. Pao and J. W. C. III. A solution to the syntactical induction-inference problem for regular languages. *Computer Languages*, 3:53–64, 1978.

J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, Calif., 1993.

L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–16, January 1986.

E. Riloff and W. Lehnert. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems*, pages 296–333, July 1994.

E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049, 1996.

R. L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs, N.J., 1971.

H. Rulot and E. Vidal. An efficient algorithm for the inference of circuit-free automata. In G. A. Ferrate, editor, *Syntactic and Structural Pattern Recognition*. Springer-Verlag, Berlin, 1988.

G. Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs, N.J., 1971.

C. Schaffer. Selecting a classification method by cross-validation. *Machine Learning*, 13(1):135–143, October 1993.

C. Schaffer. A conservation law for generalization performance. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1994.

D. Sleator and D. Temperley. Parsing English with a link grammar. *Third International Workshop on Parsing Technologies*, 1993.

S. Soderland and W. Lehnert. Wrap-Up: a trainable discourse module for information extraction. *Journal of Artificial Intelligence Research*, 2:131–158, 1994.

S. Soderland. *Learning Text Analysis Rules for Domain-specific Natural Language Processing*. PhD thesis, University of Massachusetts, 1996. CS Tech. Report 96-087.

S. Soderland. Learning to extract text-based information from the world wide web. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 1997.

A. Stolcke and S. M. Omohundro. Best-first model merging for hidden Markov induction. Technical Report TR-94-003, International Computer Science Institute, Berkeley, California, January 1994.

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, Inc., Boston, 1979.

E. Vidal. Grammatical inference: an introductory survey. In R. C. Carrasco and J. Oncina, editors, *Grammatical Inference and Applications: Second International Colloquium, ICGI-94*, pages 1–4. Springer-Verlag, September 1994.

R. Weischedel, D. Ayuso, S. Boisen, H. Fox, R. Ingria, T. Matsukawa, C. Papageorgiou, D. MacLaughlin, M. Kitagawa, T. Sakai, J. Abe, H. Hosihi, Y. Miyamoto, and S. Miller. BBS: description of the PLUM system as used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pages 93–107, August 1993.

C. A. Will. Comparing human and machine performance for natural language information extraction: results for English microelectronics from the MUC-5 evaluation. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, August 1993.

D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.