A Maximum Entropy Approach to Named Entity Recognition

by

Andrew Borthwick

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Computer Science Department
New York University

September, 1999

Dissertation Advisor

For my wife, Sarah

Acknowledgments

This work would not have been possible without the support of many people inside and outside of New York University.

My advisor, Professor Ralph Grishman, has provided me with a great deal of useful advice, including suggesting the problem of named entity recognition to me as a promising application for maximum entropy modeling. More than that, he has helped me work through a great deal of literature in statistical computational linguistics and he generously supplied me with the necessary time, equipment, and resources of his research staff which enabled me to put together the MENE system.

I would also like to thank the other members of NYU's Proteus project for their assistance. In particular, John Sterling helped me to develop the idea of integrating the Proteus parser with the MENE system in the month before the MUC-7 evaluation. He and Eugene Agichtein put in extremely long hours leading up to the evaluation and helped to make it a success.

The work on porting the MENE system to Japanese would not have been possible without the assistance of my friend and colleague, Satoshi Sekine. In addition, I would like to thank him for helping me out as the only English-speaking participant in the IREX evaluation. For his assistance with my upcoming trip to Japan and for all his work on translating IREX instructions for my benefit, I am very grateful.

I would like to thank Professor Ernie Davis for agreeing to be on my thesis committee and for his advice and support as a teacher and as an advisor in my first years as a student. I would also like to thank Professor Ravi Boppana. His skillful teaching and patient tutoring were immensely helpful to me in my studying for the departmental exam in theoretical computer science.

Outside of NYU, I would like to thank Erik Ristad for his Maximum Entropy Modeling Toolkit, which I used in this work and for his support of the system by supplying me with debugging help and various system updates. My thanks also go to Dekang Lin of the University of Manitoba and George Krupka of IsoQuest for collaborating with me by allowing me to test my system on their systems' outputs.

My understanding of maximum entropy modeling, computational linguistics, and object oriented programming was greatly broadened and deepened by the internship which I spent at IBM in the summer of 1997. In particular, I would like to thank Kishore Papineni and Radu Florian, with whom I worked on the design and implementation of a very interesting maximum entropy language modeling system. I would also like to thank Mark Epstein, Harry Printz, and Todd Ward, who all were of great assistance to me that summer. Finally, I would like to thank Salim Roukos, for assigning me to such an educational project, for his encouragement throughout the internship, and for agreeing to serve on my thesis committee.

In closing, I would like to thank my wife, Sarah, for putting up with having a

graduate student as a boyfriend, fiance, and then husband for all these long years. For her tolerance of all the late nights and weekends which I have put in studying or working on experiments, I am grateful. I would also like to thank our one-year-old daughter, Phoebe, for supplying me with insights into limited-vocabulary linguistics: “duck”, “no”, and “Da-da”.

Abstract

This thesis describes a novel statistical named-entity (i.e. "proper name") recognition system known as "MENE" (Maximum Entropy Named Entity). Named entity (N.E.) recognition is a form of information extraction in which we seek to classify every word in a document as being a person-name, organization, location, date, time, monetary value, percentage, or "none of the above". The task has particular significance for Internet search engines, machine translation, the automatic indexing of documents, and as a foundation for work on more complex information extraction tasks.

Two of the most significant problems facing the constructor of a named entity system are the questions of portability and system performance. A practical N.E. system will need to be ported frequently to new bodies of text and even to new languages. The challenge is to build a system which can be ported with minimal expense (in particular minimal programming by a computational linguist) while maintaining a high degree of accuracy in the new domains or languages.

MENE attempts to address these issues through the use of maximum entropy probabilistic modeling. It utilizes a very flexible object-based architecture which allows it to make use of a broad range of knowledge sources in making its tagging decisions. In the DARPA-sponsored MUC-7 named entity evaluation, the system displayed an accuracy rate which was well-above the median, demonstrating that it can achieve the performance goal. In addition, we demonstrate that the system can be used as a post-processing tool to enhance the output of a hand-coded named entity recognizer through experiments in which MENE improved on the performance of N.E. systems from three different sites. Furthermore, when all three external recognizers are combined under MENE, we are able to achieve very strong results which, in some cases, appear to be competitive with human performance.

Finally, we demonstrate the trans-lingual portability of the system. We ported the system to two Japanese-language named entity tasks, one of which involved a new named entity category, "artifact". Our results on these tasks were competitive with the best systems built by native Japanese speakers despite the fact that the author speaks no Japanese.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The subject of this work is a novel system for tagging named entities in text called "MENE", an acronym which stands for "Maximum Entropy Named Entity". This chapter will describe the problem of named entity recognition, discuss its importance, and look at how named entity systems are currently being evaluated.

1.1 Information Extraction

Named entity recognition (which might also be called "proper name classification") is a computational linguistics task in which we seek to classify every word in a document as falling into one of eight categories: person, location, organization, date, time, percentage, monetary value, and "none-of-the-above". In the taxonomy of computational linguistics tasks, it falls under the domain of "information extraction". Information extraction is the task of extracting specific kinds of information from documents as opposed to the more general task of "document understanding" which seeks to extract all of the information found in a document.

There are many levels of sophistication which one can attempt in information extraction. The most ambitious task currently being widely attempted is "scenario template" extraction. In this task, we seek to retrieve a wide variety of information about a certain type of event from a document. For instance, at the recent "Seventh Message Understanding Conference" (MUC-7) [32], the specific scenario-template task was to identify missile and rocket launch events in 100 articles from the New York Times. Participants in this task were asked to fill in as many slots as possible in a database template to answer questions such as:

- Where was the rocket launched from?
- Who owned the rocket?
- Who owned the payload?

- What was the payload?

A simpler information extraction task is that of "template relationships". Here the task is to find the relationship between pairs of named entities. For instance, from the phrase "Microsoft president Bill Gates", we would want the system to report that "Bill Gates" is a person, "Microsoft" is an organization, and that Bill Gates is an employee of Microsoft.

1.2 Named Entity Recognition

Named entity recognition, which is the subject of this thesis, is much simpler than either of these tasks, but it is a necessary precursor to them. Clearly before we can determine the relationship between Microsoft and Bill Gates, we must first properly categorize them respectively as an organization and a person. Similarly, Cape Kennedy must first be identified as a location before we can identify it as a rocket's launching site.

Since the named entity task is relatively simple, a relatively high accuracy rate is expected of N.E. systems. While it is, indeed, fairly easy to build a named entity (N.E.) system which has reasonable performance, there are still a large number of ambiguous cases which make it difficult to attain human performance levels on the task. For instance:

- When is the word "Washington" being used as the name of a person and when as the name of a city?
- "Mr. Jones lost 25 pounds ..." Did he lose 25 pounds of weight or 25 pounds of British currency?

An example may prove helpful here. Given the paragraph:

> Italy's business world was rocked by the announcement last Thursday that Mr. Verdi would leave his job as vice-president of Music Masters of Milan, Inc. to become operations director of Arthur Andersen.

we want to reproduce the paragraph with all of the named entities marked with SGML tags as follows:

⟨ENAMEX TYPE="LOCATION"⟩Italy⟨/ENAMEX⟩'s business world was rocked by the announcement ⟨TIMEX TYPE="DATE"⟩last Thursday⟨/TIMEX⟩ that Mr. ⟨ENAMEX TYPE="PERSON"⟩Verdi⟨/ENAMEX⟩ would leave his job as

vice-president of ⟨ENAMEX TYPE="ORGANIZATION"⟩Music Masters of Milan, Inc.⟨/ENAMEX⟩ to become operations director of ⟨ENAMEX TYPE = "ORGANIZATION" ⟩Arthur Andersen⟨/ENAMEX⟩.

Note a few difficult cases in this paragraph:

- "Italy" is at the beginning of a sentence, so capitalization information is useless.
- The "'s" is not part of the name "Italy"
- The date is "last Thursday" rather than "Thursday"
- "Milan" is tagged as a part of an organization name rather than as a location
- "Arthur Andersen" is an organization, not a person

These cases suggest some more general questions of robustness and portability which this thesis will attempt to addresss, such as:

- How can a system recognize names when they appear in headlines or at the beginning of sentences and capitalization information is consequently missing?
- Does a system have to be rewritten whenever there is a shift in domain or language?
- What happens if the rules are changed a little bit? For instance, according to the rules of the MUC-7 evaluation under which we tested our system the word "Ford" in "Ford Taurus" would not be tagged as an organization because it is seen as being part of a product name. However, another user might want this tagged as a reference to the company.

1.3 Applications of Named Entity Recognition

There has been a considerable amount of work on named entity taggers in recent years which aims to address many of these ambiguity and portability issues and at least one company has been built around providing a solution for this problem (IsoQuest, Inc.) [29]. This interest has been largely motivated by the relative tractibility of the problem and the potential marketability of an accurate named entity system. This marketability is driven by the many obvious benefits of having an accurate named entity system, including:

- More accurate internet search engines. One could find references to Clinton, South Carolina, without wading through pages of information about President Clinton, for instance.
- General document organization. A user can call up all documents on a company intranet which mention a particular individual.
- Before reading an article a user could see a list of the people, places, and companies mentioned in the document.
- Automatic indexing of books. For many books, the majority of the items which would go in the index would be named entities.
- *People Magazine* could use this to highlight the names of every person mentioned in **bold**. *The Wall Street Journal* could do the same with companies.
- A named entity tagger can serve as a preprocessing step to simplify tasks such as machine translation.
- As mentioned earlier, an N.E. tagger is an essential component of more complex information extraction tasks.

1.4 Evaluating Named Entity Systems

Recent progress in English N.E. has been greatly facilitated by the MUC-6 and MUC-7 machine understanding conferences. These conferences, organized by the Defense Advanced Research Projects Agency (DARPA) have as their primary purpose the evaluation of information extraction systems in a carefully controlled test followed by a conference in which participants present papers discussing their methods.

In Japanese, the "Multilingual Entity Task" (MET-2) followed the same basic format as the MUC evaluations for Japanese named entity identifications, making use of the same domains and with results being presented at the same conference. The "Information Retreival Exercise" (IREX) is a similar Japanese-language task. The evaluation was held on May 13, 1999 with a conference to be held August 30 - September 3, 1999 in Tokyo.

NYU's "Proteus" project entered the MENE system in the MUC-7 named entity evaluation [9], where it was evaluated against systems from 11 other institutions. We also entered a Japanese-language version of the system in the IREX evaluation and tested it against data from the MET-2 evaluation. Since these evaluations served as our primary test of the system's effectiveness, it is important

that we describe them in detail. We will discuss the MUC-7 evaluation here and will discuss details of the MET-2 and IREX evaluations in chapter 8.

MUC-7 was a carefully controlled test consisting of four stages.

1. Training data was distributed to the participants consisting of 100 articles on the subject of aviation disasters. These articles consisted of about 111,000 words total.

2. A "dry run" test was conducted

   (a) A "dry run" test corpus of 25 articles consisting of about 25,000 words was distributed.

   (b) Sites participating in this dry run test ran their systems against the 25 articles. Sites were instructed that the dry run test data should not be read by either the participants or their systems prior to this test run.

   (c) Participating sites sent in the output of their systems on the dry run test data. This output consisted of an exact copy of the test corpus except that the participating N.E. system would insert SGML markings into the text to bracket the named entities which it identified. Examples of this marking can be found on page 2 and appendix B.

   (d) System administrators scored every system's output against a master key using an automated scoring program.

3. A revised training corpus was distributed which had various minor tagging errors corrected.

4. The formal run evaluation was held. This evaluation followed the same format as the dry run except that it was conducted on 100 articles and the subject matter shifted from aviation disasters to missile and rocket launches. This shift in test domain was not communicated to the participants beforehand and had some significant implications for system performance. An example of MENE's output on one of these formal run articles can be found in appendix B.

Note an important caveat here. Systems were allowed to supplement the official training data of steps 1 and 3 with data which they had tagged themselves or acquired from other sites. We trained the final MENE system on 250 supplementary articles tagged by NYU, BBN [37], and the conference organizers in addition to the 100 official articles for a total of 350 articles (321,000 words) of training data.

Individual named entities were scored based on whether they had the correct start and end points (i.e. were any words left out of the name?) and whether

$$REC = \frac{correct}{correct + incorrect + missing} \quad (1.1)$$

$$PRE = \frac{correct}{correct + incorrect + spurious} \quad (1.2)$$

$$F = \frac{2 \cdot PRE \cdot REC}{PRE + REC} \quad (1.3)$$

Figure 1.1: F-measure formula

they identified the entity correctly (i.e. "person" vs. "organization"). These were called "text" and "type" factors, respectively. Note that the text of a given tag could be correct while the type could be incorrect. The reverse is also possible, such as when a tag has an incorrect start or end point but is correctly labeled for the words which it does cover. Hence each tag had the potential to be scored "correct" twice, once for tag and once for type.

System scores for each test were measured in terms of precision, recall, and "F-measure" which were computed as follows [11]. First, given a tagging by an N.E. system (a "response") and an answer key which has the correct taggings, define the quantities "correct", "incorrect", "missing", and "spurious" as the number of instances of the quantities defined in table 1.1.

correct	response equals key
incorrect	response not equal to key
missing	key is tagged, response is untagged
spurious	response is tagged, key is untagged

Table 1.1: Definitions used in F-measure computation

These quantities are then used to compute precision, recall, and F-measure as shown in figure 1.1.

At the MUC-6 and MUC-7 conference, systems were judged based on their F-measures. Consequently, this is the score which we will be using in this work to judge the quality of our named entity systems.

Chapter 2

Prior Work in Named Entity Recognition

The MENE system uses a very flexible, maximum entropy approach to named entity recognition. This chapter will look at systems from NYU, BBN, and other institutions which take different approaches to the problem. We will also look at a system from the University of Edinburgh, which, along with our MENE system, was the first to make use of maximum entropy in named entity recognition.

2.1 The Handcrafted Approach

The majority of the systems participating in MUC-7 used what could broadly be described as a "handcrafted approach". By this we mean that these are systems which are built by hand and rely heavily on the intuition of their human designers.

NYU's "Proteus" named entity system [25] typifies this approach to the problem. This system, which was NYU's entrant in the MUC-6 N.E. evaluation, is written in Lisp and is primarily composed of a large number of context-sensitive reduction rules. These rules are mostly very intuitive, the sorts of rules which immediately leap to mind when one thinks about how an N.E. system might be built. For instance, below we have a few such rules from Proteus with examples in which they are right and wrong.

- Title Capitalized_Word $\Longrightarrow$ Title Person_name
  - Correct: Mr. Jones, Gen. Schwarzkopf
  - Incorrect: Mrs. Field's Cookies (A corporation), Mr. Ten-Percent (nickname for a corrupt third-world official)
- Month_name number_less_than_32 $\Longrightarrow$ Date

  - Correct: February 28, July 15
  - Incorrect: Long March 3 (a Chinese Rocket)

- from Date to Date $\Longrightarrow$ Date
  - Correct: from August 3 to August 9, 1997
  - Incorrect: We moved the conference from April to June to allow more time for preparation (April and June should be tagged separately, not tagged as the single date "from April to June").

While some of the examples which cause these rules to fail might seem far-fetched, the "Long March 3" example did appear in the MUC-7 formal evaluation corpus (there are two instances of "Long March" in the walk-through article). In fact, for almost any named-entity rule there will be numerous exceptions. It is generally impossible, given the usual time constraints, even to code for every exception which one can think of, leaving aside those exceptions which don't become apparent until one has run a test.

In addition, every different type of document will have its own idiosyncrasies. For instance, in the New York Times articles which made up our test and training corpora, the beginning of the main body of each article tended to be very stereotyped, such as this excerpt from the formal run:

> Bethesda, Maryland, Feb. 15 (Bloomberg) – Comsat Corp. and the U.S. government proposed a restructuring of ...

A rule which recognized this pattern at the start of a paragraph would doubtless improve the score significantly on this domain, but it is questionable whether it would carry forward into a new domain.

Another serious issue with the handcrafted approach is that of expense. Just getting a system up and running requires several person-months of time from a programmer with significant experience in computational linguistics–a scarce commodity. This time is largely wasted if we want to then port the system to a new language or domain.

There is also a serious question of consistency and reproducibility of results. At the MUC-7 conference, the second-ranked named entity system, from IsoQuest [29], got an F-measure of 91.6, while the lowest-ranked handcoded N.E. system from an English-speaking institution (the European FACILE Consortium) got F=81.91 [6]. Interestingly, the IsoQuest and the FACILE systems both seem somewhat similar. Both rely on handcoded rules. Both make use of databases of common named entities. Both also allow different weights to be assigned to the rules so that conflicts between rules predicting different entities can be resolved by choosing the rules which have the greatest weight.

The major difference between the two systems seems to be that IsoQuest's has had substantially more person-months invested in it. FACILE states that they only invested one person-month in developing and testing the linguistic resources for MUC-7 as opposed to developing the underlying software. IsoQuest states that they, too, devoted only about one month to customizing their system for MUC-7, but they were building their system on top of their basic commercial system. Consequently, 90% of the patterns in the MUC-7 entry came from their commercial system. Since they say that the commercial system has been under development for about two years and has been licensed to over 30 clients, one would expect that its basic patterns are quite good. Furthermore, they have built a slick GUI development environment, which probably further leverages their development efforts.

In sum, if one is smart enough and works hard enough, it is possible to build a strong named entity system using conventional handcoded techniques. However, these systems will still have a number of drawbacks

1. They will be expensive, since they will rely on the expertise of trained computational linguists.

2. They will have to be manually adapted to new domains

3. Their rules and lexicons must be completely rewritten when they are ported to new languages

4. Performance will be highly sensitive to the computational linguist's skill in writing the named entity patterns and to the amount of labor devoted to the task.

On the other hand, there are certain classes of patterns which are difficult to capture except through the use of regular expressions. For instance, the hypothetical pattern [person_name, "the" adjective* "CEO of" organization] which correctly covers a phrase like "Fred Smith, the young dynamic CEO of XYZ Enterprises" is a pattern which would be difficult for an automated system to learn because of the presence of the sequence of zero or more adjectives. As we will discuss later on, our official entry in the MUC-7 evaluation tried to combine the best of both worlds by allowing the MENE system to look at the output of the Proteus system as one of its inputs.

2.2 Automated Approaches: Training Data

It is clear that to answer these objections to the handcoded systems one must attempt to build a system which will in some way automatically train itself, thereby cutting the slow and expensive computational linguist out of the development loop.

There are, however, a large number of ways of building such a system, so we will be looking at three of them in turn, starting with Sekine's decision-tree based approach [48].

Like all of the automated methods which we will be discussing in this chapter, the decision-tree method takes as its starting point a body of text from the target domain which has been been human-annotated with all the "correct" named entities. This training text is in essentially the same form as the output which the systems are supposed to generate. An example of this marking was shown on page 2.

Clearly the creation of large amounts of training text is a burden which is not placed on the handcrafted systems. On the other hand, this work does not require the quantity of labor which the handcrafted systems require. This author has received reports that 100 articles (roughly 100,000 words) of text can be tagged in between one [48] and three [49] person-days. This compares with the person-month required to code the rules of even the poorest-performing named entity system.

Furthermore, text-tagging does not require the use of highly-trained computational linguists. BBN [37] made use of a team of undergraduates to tag 700,000 words of data for their system. A high rate of accuracy can be maintained by having two different annotators tag each piece of text and then using a third annotator to resolve any disputes.

One final point to make on training data is that even a hand-crafted system typically needs at least a small annotated corpus for testing purposes. This data is used to test the system and to prime the intuition of the system designer.

2.3 Automated Approaches: Decision Trees

Before looking at the decision-tree method in detail, we first have to look at some general characteristics of how one can abstract the problem of named entity recognition into a mathematically tractable form.

Given a tokenization of a test corpus and a set of n (for MUC-7, $n = 7$) named entity categories, the problem of named entity recognition can be reduced to the problem of assigning one of $4n + 1$ tags to each token. For any particular N.E. category x from the set of n categories, we could be in one of 4 states: x_start, x_continue, x_end, and x_unique. In addition, a token could be tagged as "other" to indicate that it is not part of a named entity. For instance, we would tag the phrase [Jerry Lee Lewis flew to Paris] as [person_start, person_continue, person_end, other, other, location_unique].

A decision tree can be considered to be composed of three elements [31]:

- Future: The possible outputs of the decision tree model. For instance, in our case the 29 different tags described above form the space of futures.

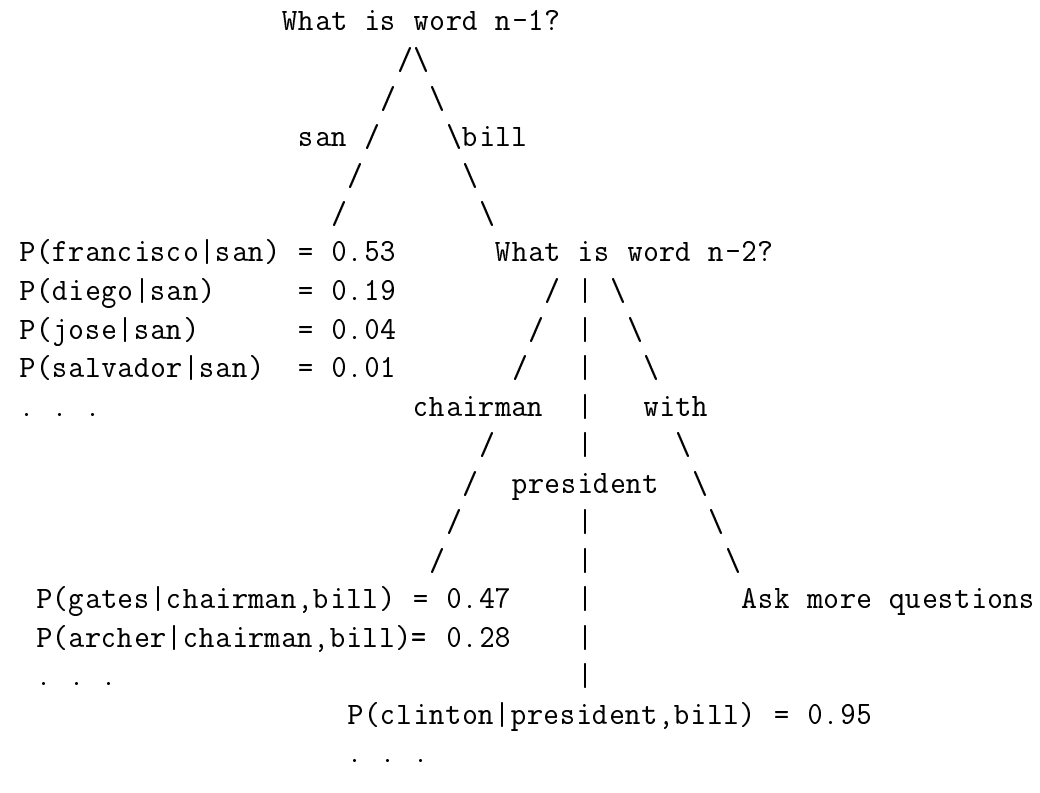

Figure 2.1: Decision Tree Example

- History: The information available to the model. In Sekine's decision tree model, this was information derivable from the previous, current, and following word, although in principal, there is no reason why the window could not have been widened.

- Questions: This is what a decision tree is all about. The objective of the decision tree growing algorithm is to find the best sequence of questions to ask about the history to determine the future. Note that in determining this sequence of questions, the choice of the mth question to ask is determined by the answers to the previous $m-1$ questions.

Once built, a decision tree is very easy to use. For instance, consider a language model which attempts to determine the next word in the text given a "history" consisting of the two previous words. A decision tree implementation might look like figure 2.1.

Once we have answered the questions and worked our way from the "root" to the "leaves" (terms which have the same meaning in decision trees as they do in the trees of algorithmic computer science), we take the probability distribution

stored at the leaf node as defining the function $P(v_i|v_{i-2}v_{i-1})$

While it should be clear from the above that decision trees offer the possibility of building a language model (or a named entity tagger) which is potentially efficient in the use of both time and space at run-time, the question we are left with is how these trees are to be built.

The basic idea is that we seek to build a tree which at every point asks the question W which reduces uncertainty about the set of futures, $\mathcal{F}$, by the greatest amount [1] . Uncertainty is measured here as conditional entropy:

$$\mathcal{W} \equiv \{\text{All possible answers to question } W\} \tag{2.1}$$

$$H(F|W) = -\sum_{q \in \mathcal{W}} P(q) \sum_{f \in \mathcal{F}} P(f|q) \log P(f|q) \tag{2.2}$$

The problem with the method as we have sketched it thus far is that in looking for the best question at any given tree node, we will tend to choose those questions W for which $|\mathcal{W}|$ is large since in general those questions will lead to the lowest value of $H(F|W)$. But while our metric will tend to make us favor large values of $|\mathcal{W}|$, from a computational point of view we want to avoid unnecessary data fragmentation and so we would prefer smaller $|\mathcal{W}|$.

The solution to this is to ask only binary questions, i.e. questions whose answer can only be "yes" or "no". This makes the problem somewhat similar to the old television game show "What's My Line", in which contestants attempted to determine the profession of a guest by asking a series of yes/no questions. In playing this game, just like in seeking to minimize $H(F|W)$, we are basically looking for questions which will produce a roughly even split between yes and no answers and in which the universe of professions for the two answers are markedly different. Hence a question like "Are you a plumber?" is a poor choice for a first question because while $P(plumber|yes) = 1$, $P(yes)$ is low. On the other hand, a question like "Do you use Colgate toothpaste?" divides the population roughly evenly, but it probably would not reduce uncertainty about the guest's profession. A better question to start with might be "Do you have any post-secondary education?", a query which does a fairly good job on both counts.

Growing a tree by selecting the questions which lead to the greatest reduction in conditional entropy is a well-known technique, and, in fact, Sekine was able to use an off-the-shelf toolkit [39] to grow his decision trees. The critical question, then, is in supplying the tree with a sufficiently rich history so that it can ask a series of informative questions which can reduce the uncertainty about the space of futures. We will now examine the specific information used by Sekine's Japanese named entity tagger.

[1] Note that in this section a capital letter (F) refers to a variable in the usual mathematical sense, a lower-case letter (f) refers to an instantiation of the variable, and a calligraphic variable ($\mathcal{F}$) is the set of all instantiations which the variable can take.

Named entity tagging in Japanese poses a particular problem in that the language has no spacing between individual words. In order to get around this problem and to focus on N.E. rather than on word segmentation, Sekine used an off-the-shelf segmenter called "Juman" [34]. In addition to doing word segmentation, Juman also returns the character type (i.e. Katakana, Kanji, etc.) of every word and tries to determine the word's part-of-speech.

The other resource Sekine provided to his system was a set of dictionaries (word-lists), which gave common words which could be expected to appear as the prefix or suffix of a named entity as well as a list of words which could be the N.E. itself. For instance, the suffix "san" would appear in the person-suffix dictionary because it is indicative of a name appearing in the prior word (e.g. "Sekine-san"). Likewise, a Japanese person name like "Sekine" would go in the person dictionary.

The approach resulted in some significant successes. The system turned in competitive results at MET-2 [46] (the Japanese version of MUC-7) and it was highly portable. In particular, it was simple to add a new type of named entity ("position", i.e. "President", "Professor"), and the system ported easily between an airline disaster domain and a business management succession domain.

One drawback to the approach, though, became apparent when the system was compared side-by-side with MENE on the MET-2 corpus. We found that when the two systems were trained on the same inputs (i.e. the "features" of the maximum entropy system had access to the same information as the "questions" of the decision tree system), they got similar results. However, one piece of information which was notably absent from the decision-tree system was the words themselves (as opposed to information about the words' part of speech, character-type, presence in various dictionaries, etc.). Integrating lexical information into this sort of decision-tree is not a trivial task since if one has a vocabulary of size n and one allows n questions of the form "Is the current word xyz?", then one is likely to harm the model by causing excessive fragmentation of the training corpus. In other words, each leaf node would have too few events to properly estimate the probabilities for the different named entities, and accuracy would suffer. Some unpublished experiments confirmed that lexical information could not be integrated into the model by such a naive approach [45]. There are more sophisticated ways of adding lexical information to a decision tree model [31], so it remains to be seen whether a decision tree N.E. system could get a performance boost out of this type of information.

As we will discuss later, lexical information can be added to MENE using almost trivial methods. When we added lexical information to the Japanese MENE system, we saw that the F-measure increased by over 3.8 F-measures, an improvement which caused it to substantially outperform the decision tree system on MET-2 data, as discussed in section 8.6.

2.4 Automated Approaches: Hidden Markov Models

A second automated approach which has been advanced recently is the use of Hidden Markov Models at BBN in their Identifinder system [5] [37]. BBN's essential idea is to build a separate bigram language model for each name category. In addition, they build a model which predicts the next name category based on the previous word and previous name category.

Thus, a simplified version of their system turns on the following two equations:

$$P(NC|NC_{-1}, w_{-1}) = \frac{c(NC, NC_{-1}, w_{-1})}{c(NC_{-1}, w_{-1})} \tag{2.3}$$

$$P(\langle w, f\rangle|\langle w, f\rangle_{-1}, NC) = \frac{c(\langle w, f\rangle, \langle w, f\rangle_{-1}, NC)}{c(\langle w, f\rangle_{-1}, NC)} \tag{2.4}$$

Some definitions of terms used in BBN's equations are as follows:

$$NC \equiv \text{Current name class} \tag{2.5}$$

$$NC_{-x} \equiv \text{Name class of the word } x \text{ words back} \tag{2.6}$$

$$c(W) \equiv \text{Count of } W. \text{ i.e. number of times event } W \text{ appears in the training corpus} \tag{2.7}$$

$$w \equiv \text{A word} \tag{2.8}$$

$$f \equiv \text{A feature} \tag{2.9}$$

Note that features in the BBN system, unlike features in maximum entropy, are attributes of words. For instance, a word can have the feature that it is capitalized, that it is at the start of a sentence, or that it is a member of a particular word list (such as a list of corporate designators).

The idea behind these equations is that we have a model predicting the next name class given the previous name class and previous word and a model predicting the next word-feature pair given the previous word-feature pair and the current name class. We then do a Viterbi search to find the sequence of name classes which assigns the highest probability to the test corpus. This name class sequence implies a tagging of the test corpus.

As an example, given appropriate training data, we could expect the following inequality to hold:

$$P(\langle\text{andersen,capitalized}\rangle|\langle\text{arthur,capitalized}\rangle_{-1}, \text{organization\_name}) > P(\langle\text{andersen,capitalized}\rangle|\langle\text{arthur,capitalized}\rangle_{-1}, \text{person\_name})$$

Consequently, we could expect that the Viterbi search routine would choose "organization name" for this sequence rather than "person name".

BBN's system had considerable success at MUC-7. They ended up with an F-measure of 90.44 and a third-of-twelve ranking, which exceeded MENE's 88.80 F-measure and fourth-place ranking. We will compare the system results in further detail in section 7.5, but for the moment, we would like to point out what we perceive to be a shortcoming of the HMM method relative to our maximum entropy approach.

We would argue that Identifinder suffers from the fact that it makes heavy use of "back-off" in its modeling. Specifically, in the event that it has never seen the exact combination of words and features described by equation 2.4, Identifinder backs off to a sequence of less specific models in the following sequence:

$$P(\langle w, f\rangle|NC) \tag{2.10}$$

$$P(w|NC) \cdot P(f|NC) \tag{2.11}$$

$$\frac{1}{|V|} \cdot \frac{1}{\text{number of word features}} \tag{2.12}$$

Note that any backoff strategy requires a method of splitting the probability "mass" between each level of backoff. In [5], BBN gives the formula which they use to do this. Also note that in addition to the above backoff sequence, there is a separate backoff sequence which is followed when an unknown word is encountered either as the current word, as the previous word, or as both.

All of this backing off exacts a certain price. Firstly, there is a question of the complexity of the model as more layers of backing off are introduced. Secondly, the issue of how the different layers of backoff are weighted against each other becomes crucial. While BBN's is a reasonable approach, the choice of method will have a major impact on the outcome.

Most importantly, though, this sort of model is vulnerable to the same sorts of data fragmentation issues which afflict the decision tree model. One needs to be careful not to introduce too many features into the model because that would increase the frequency with which the system would have to back off. One should also note that Identifinder only allows one feature to be active at a time. Consequently, the system would have a difficult time modeling a situation where a word was, for instance, both capitalized and on a list of names of months of the year, a situation which might help to distinguish words like "march" and "may".

Granted, these criticisms must be taken with a grain of salt since we have not yet demonstrated that MENE can outperform an HMM system when the two systems are placed on an equal footing. However, we feel that the fact that the maximum entropy approach doesn't suffer from these problems points to the long-term potential of the approach.

2.5 A Hybrid Approach: LTG/University of Edinburgh

The other interesting statistical system entered in the MUC-7 evaluation was, like MENE, a hybrid statistical-handcoded system which made use of maximum entropy [35]. The key characteristic of this system is that the processing is done in stages. In the initial phase, the text passes through some "sure-fire" handcoded regular expression rules like those described in the Proteus system. These are rules which were deemed to have a very high probability of being correct (and which, in fact, performed with 98% precision on the formal evaluation). One example of these is the following:

- Capitalized_word^{+} is a? JJ* PROF
  - Example: Yuri Gromov is a former director
  - Definitions:
    - "+" means "one or more"
    - "*" means "zero or more"
    - "?" means "zero or one"
    - "JJ" means adjective
    - "PROF" is a profession (director, manager, analyst, etc.)

Since these sure-fire rules have only 42% recall, some additional stages are necessary. In the next stage, a set of weaker rules are passed to a maximum entropy model. These rules take into account information such as whether or not a word was identified as an N.E. elsewhere in the text by one of the sure-fire rules, case information, the position of the word in the sentence, etc. Although the LTG paper gives very little information about their M.E. model, one imagines that each of the above might be a different feature which would be given a weight by a maximum entropy training procedure similar to the MENE procedure described in section 4.2.

Following this maximum entropy phase, there is another stage in which handcoded rules are used. These rules have more relaxed criteria than the "sure-fire" rules and consequently they have lower precision. These rules make extensive use of lists of known locations, organizations, and person-names. There follows another maximum entropy stage similar to the one described above and finally another M.E. model which handles names found in the document headers (i.e. in headlines).

Unfortunately, comparisons between our work and that of LTG are difficult since over half of the LTG system's recall came from the two non-statistical phases

of their five-stage process. The LTG system demonstrated superior performance on the formal run relative to the MENE-Proteus hybrid system (93.39 vs 88.80) and, in fact, had the highest score overall at MUC-7, but it isn't clear whether their advantage came from superior handcoded rules or superior statistical techniques, because their system is not as easily broken down into separate components as is MENE-Proteus. It is also possible that tighter system integration between the statistical and handcoded components was responsible for some of LTG's relative advantage, but note that MENE-Proteus appears to have an advantage over LTG in terms of portability. As we will discuss later, we were able to easily port MENE to Japanese, and we expect that it could be easily combined with a pre-existing Japanese handcoded system, but it isn't clear that this could be done with the LTG system. Nevertheless, one avenue for future research is to look at tighter multi-system integration methods which wouldn't compromise MENE's essential portability.

Chapter 3

Maximum Entropy Modeling

Maximum entropy is a very flexible method of statistical modeling which turns on the notion of "futures", "histories", and "features". Futures are defined as the possible outputs of the model. Like in the decision tree method, we are seeking to choose from the set of 29 different named entity tags described in section 2.3. A maximum entropy solution to this, or any other similar problem allows the computation of $p(f|h)$ for any f from the space of possible futures, F, for every h from the space of possible histories, H. As with decision trees, a "history" in maximum entropy is all of the conditioning data which enables you to assign probabilities to the space of futures. In the named entity problem, we could reformulate this in terms of finding the probability of f associated with the token at index t in the test corpus as:

$$p(f|h_t) = p\left(f|\begin{array}{l}\text{Information derivable from the test corpus}\\ \text{relative to token } t\end{array}\right) \tag{3.1}$$

The computation of $p(f|h)$ in M.E. is dependent on a set of "features" which, hopefully, are helpful in making a prediction about the future. Like most current M.E. modeling efforts in computational linguistics, we restrict ourselves to features which are binary functions of the history and future. For instance, one of our features is

$$g(h,f) = \left\{ \begin{array}{lcl} 1 & : & \begin{array}{l}\text{if current-token-capitalized(h) = true and}\\ f = \text{location\_start}\end{array}\\ 0 & : & \text{else} \end{array} \right\} \tag{3.2}$$

Here "current-token-capitalized(h)" is a binary function which returns true if the "current token" of the history h (the token whose tag we are trying to determine) has an initial capitalized letter. Clearly when current-token-capitalized(h) holds, the current word is more likely to be the start of a location name than average. The question, as was the case in the previous chapter, is how to turn these intuitively appealing features into a conditional probability.

Given a set of features and some training data, the maximum entropy estimation process produces a model in which every feature g_i has associated with it a parameter α_i. This allows us to compute the conditional probability as follows [4]:

$$P(f|h) = \frac{\prod_i \alpha_i^{g_i(h,f)}}{Z_\alpha(h)} \tag{3.3}$$

$$Z_\alpha(h) = \sum_f \prod_i \alpha_i^{g_i(h,f)} \tag{3.4}$$

To rephrase, this equation tells us that the conditional probability of the future given the history is the product of the weightings for all features which are active on the $\langle h, f \rangle$ pair, normalized over the products for all the futures. The following chapter will describe how we arrive at this formula as the correct way to compute $p(f|h)$ and how we can compute the weightings α_i.

One thing to bear in mind while reading the chapter is that although the mathematics behind M.E. can be quite complex, in the end the computation of the value of the parameters α_i by the M.E. estimation routine can be treated as a "black box" (we used an off-the-shelf toolkit [43]). This allows the modeler to concentrate on selecting the features which best characterize the problem while letting the M.E. estimator worry about assigning the features their relative weights.

3.1 Constraints on Probability Distributions

Maximum entropy modeling was first described by E. T. Jaynes in [26] and more recently in a draft manuscript available on the web [28]. A crucial point of Jaynes' work is that he argues for viewing modeling problems in terms of constraints on the probability distribution. For instance, consider the following equations over a training corpus C:

$$Q \equiv \left\{ \begin{array}{l} \text{An equivalence class over } \mathcal{H}. \text{ E.g. "The} \\ \text{set of } h \text{ such that } h_{t+1} \text{ is 'announced' "} \end{array} \right\} \tag{3.5}$$

$$y \equiv \text{the future "organization\_unique"} \tag{3.6}$$

$$J = \frac{|\{(h,f) \in C : h \in Q \wedge f = y\}|}{|\{(h,f) \in C : h \in Q|} \tag{3.7}$$

It is not possible to simply assign $P(y|h) = J$ over the entire equivalence class Q because there will be cases where the probability of y should be higher or lower depending on the other characteristics of h. Maximum entropy addresses this problem by allowing the conditional probability to fluctuate from this average J and only insisting that the model assign this probability J on average over the

training corpus [44]. This is expressed by the following "constraint":

$$\sum_{(h,f)\in C:h\in Q\wedge f=y} P(h)P(f|h) = J \cdot P(h \in Q) = K \tag{3.8}$$

Another way of saying this is that we require that the expected value of $P(y|h)$ over the equivalence class Q is J. Computationally, it will be convenient to replace our conditioning on Q and y with a feature or "index" function $g_r(h, f)$ as follows:

$$R \equiv \left\{ \begin{array}{l} \text{An equivalence class over } (\mathcal{H}, \mathcal{F}). \text{ I.e.} \\ \text{“}h_{t+1} = \text{‘announced’ and } f = \text{organiza-} \\ \text{tion\_unique} \end{array} \right\} \tag{3.9}$$

$$g_r(h, f) = \left\{ \begin{array}{lcl} 1 & : & \text{if } (h, f) \in R \\ 0 & : & \text{else} \end{array} \right. \tag{3.10}$$

Then we can express our constraint on the probability distribution as:

$$\sum_{(h,f)} P(h, f)g_r(h, f) = K \tag{3.11}$$

Since we are only trying to model conditional probabilities rather than joint probabilities, it makes sense to reformulate our constraint as

$$\sum_{(h,f)} \tilde{P}(h)P_{ME}(f|h)g_r(h, f) = K \tag{3.12}$$

where $\tilde{P}(h)$ is the empirical probability of h in the training corpus and of course, $P_{ME}(f|h)$ is the conditional probability assigned by our model [10]. This reformulation restricts the sum to all histories which occurred in our training corpus rather than to all possible histories, thereby making the calculation computationally tractable. Combining the computation of the value of K with this equation, we get a consolidated constraint equation of:

$$\sum_{h,f} \tilde{P}(h, f) \cdot g_r(h, f) = \sum_h \tilde{P}(h) \cdot \sum_f P_{ME}(f|h) \cdot g_r(h, f) \tag{3.13}$$

If we accept that this is a reasonable formulation of the problem, then we are left with the question of how best to formulate a model consistent with these constraints.

3.2 An Argument for the Correctness of M.E.

Suppose we have a set of constraints on a probability distribution such as the following. Note that in this work, w_0 indicates the current token, w_n indicates a

token n tokens after the current word, and w_{-n} indicates a token n tokens back

$$g_c(x,y) = \begin{cases} 1 & : \text{ if } (w_0 \text{ capitalized and } f = \text{person\_unique}) \\ 0 & : \text{ else} \end{cases} \tag{3.14}$$

$$g_s(x,y) = \begin{cases} 1 & : \text{ if } (w_1 = \text{“said” and } f = \text{person\_unique}) \\ 0 & : \text{ else} \end{cases} \tag{3.15}$$

We would then have two values k_c and k_s for the family of constraints described above. M.E. theory holds that we should build the model which has a probability distribution which is as close to uniform as possible while conforming to these constraints. There are a lot of different ways of saying this:

- The probability distribution should be the one which has the greatest degree of uncertainty, given the constraints. As Jaynes put it “it agrees with everything that is known but carefully avoids assuming everything that is not known” [27].
- The distribution should be the one which has the lowest Kullback Liebler distance from the uniform distribution [44]. Here the uniform distribution would be the one which for a future vocabulary $\mathcal{F}$, specified that $\forall f \in \mathcal{F}(P(f) = \frac{1}{|\mathcal{F}|})$ (this is also the maximum entropy distribution of a probabilistic model over $\mathcal{F}$ with no constraints). The Kullback Liebler distance measures the asymmetric distance between two probability functions, P(x) and Q(x). It is defined as
$$D(Q(x), P(x)) \equiv \sum_x Q(x) \log \frac{Q(x)}{P(x)} \tag{3.16}$$
- We should build a model which assumes a lack of higher-order interaction among the constraints [24]. In other words, if we want to specify that the constraints k_c and k_s have a special interaction, then we would have to define a new constraint $k_{c \wedge s}$.
- We should build the model with the “flattest” possible probability distribution.

The accepted measure of the concept of “the amount of uncertainty” is entropy. The entropy of a prior probability distribution P(x) is defined as

$$H(X) = -\sum_{x \in X} P(x) \log P(x) \tag{3.17}$$

and for conditional probabilities, it is the formula found in equation 2.2.

Having gone through all of these preliminaries, we are now prepared to sketch Jaynes' argument for maximum entropy as the only correct method of modeling probability distributions of this type [28]:

1. The information that is known about the problem should be expressed as constraints on the probability distribution in the form of equation 3.11. This is a very general form since the function g_r can be any binary valued function.

2. Given the task of coming up with a probability distribution which conforms to a set of constraints like this, the most reasonable thing to do is to conform to the set of constraints while building a model which is maximally uncertain about everything which it hasn't been told.

3. To maximize uncertainty, we need a formula which quantifies the entity. Jaynes gives a somewhat informal proof that the entropy formula (2.2, 3.17) is the only expression of uncertainty which meets the following desiderata:

   (a) We need a function $H_n(P(X))$ which sets up an association between "amount of uncertainty" and the real numbers. Here $n = |\mathcal{X}|$, where $\mathcal{X}$ is the vocabulary of X.

   (b) H_n is a continuous function of $P(X)$.

   (c) Suppose we have

   $$\forall x \in \mathcal{X} \left(P(x) = \frac{1}{|\mathcal{X}|} \right) \tag{3.18}$$

   then $H_n(P(X))$ must be a monotonically increasing function of n. This corresponds to the intuitive notion that more choices make for greater uncertainty.

   (d) H_n must be consistent: i.e. if there is more than one way to compute its value, then all approaches must yield the same result.

   (e) Suppose

   $$P(X) \equiv p(x_1), p(x_2), \ldots p(x_n) \tag{3.19}$$
   $$x_z \equiv x_j \vee x_k \tag{3.20}$$

   then

   $$\begin{aligned} H_n(P(X)) = {} & H_{n-1}(p(x_1), \ldots p(x_{j-1}), p(x_{j+1}), \ldots \\ & p(x_{k-1}), p(x_{k+1}), \ldots p(x_n), p(x_z)) + \\ & p(x_z) \cdot H_2 \left(\frac{p(x_j)}{p(x_z)}, \frac{p(x_k)}{p(x_z)} \right) \end{aligned} \tag{3.21}$$

   This last condition specifies how the H_n relate to each other for different n. The condition is logically justifiable by considering a situation in which we are first told that there are $n-1$ different choices over

which we can spread the probability mass. We are then told that one of these choices x_z is really composed of two different choices, x_j and x_k where, of course, $p(x_z) = p(x_j) + p(x_k)$. Then in addition to our original uncertainty, with probability $p(x_z)$ we will have the additional uncertainty about whether to choose x_j or x_k.

4. To sum this up, define $\Gamma = \{P_1(X), P_2(X), \ldots\}$, the infinite set of all probability distributions which conform to the constraints. Then choose the model

$$P_{max}(X) = \arg\max_i H(P_i(X)) \tag{3.22}$$

5. It can be shown [26] [16] [28] [4] that there is a unique solution to equation 3.22 of the form

$$\begin{aligned} P_{max}(f|h) &= e^{\sum_i \lambda_i g_i(h,f)} \frac{1}{Z_\lambda(h)} && (3.23) \\ &= \frac{\prod_i \alpha_i^{g_i(h,f)}}{Z_\alpha(h)} && (3.24) \\ \alpha_i &\equiv e^{\lambda_i} && (3.25) \\ Z_\alpha(h) &= \sum_f \prod_i \alpha_i^{g_i(h,f)} && (3.26) \end{aligned}$$

(Note that the marginal probability $P_{max}(x)$ is of the same form as the conditional probability except that $g_i(h, f)$ is replaced with $g_i(x)$ and $Z_\alpha(h)$ becomes the constant Z_α)

6. Since it is the unique function matching our criteria, P_{max} is the only correct probability distribution in this situation.

It can also be shown [4] that there is a probability model P_e of the form of equation 3.24 such that $P_e \in \Gamma$ and P_e has the following characteristics:

1. It is unique
2. It is the solution to equation 3.22, so $P_e = P_{max}$
3. Given a set G of binary functions $g(h, f)$, define Φ_G to be the set of all probability distributions of the form of equation 3.24 over the features $g_i \in G$. P_e is the member of Φ_G which maximizes the probability of the training corpus C.

Hence this new, sophisticated technique derived from information theory is really a form of the traditional statistical NLP technique of maximizing the probability of the training corpus. The only difference is that here we are told to do our search for the maximum likelihood model among models from Φ_G. This convergence of techniques is another argument for the validity of the method.

3.3 How to Build a Model With M.E.

A simple method of building an M.E. model is by the use of the Generalized Iterative Scaling Algorithm, which is guaranteed to converge to a solution in this kind of problem [18]. As applied to a conditional model, the algorithm is:

3.3.1 Generalized Iterative Scaling

Given a family of index functions g_i and associated expectations for the value of these functions K_i, each iteration j of this algorithm creates a new estimate of the parameters α_i which matches the constraints better than its predecessors. Each iteration consists of the following steps:

1. Compute the expectations of all the g_i's under the current estimate of the probability function. Namely, compute

$$K_i^{(j)} \equiv \sum_h \tilde{P}(h) \cdot \sum_f P_j(f|h) \cdot g_i(h,f) \tag{3.27}$$

2. Compare the *actual* values $K_i^{(j)}$ to the *desired* values K_i, and update the α_i's according to the following formula:

$$\alpha_i^{(j+1)} = \alpha_i^{(j)} \cdot \frac{K_i}{K_i^{(j)}} \tag{3.28}$$

3. Define the next estimate of the probability function based on the new α_i's:

$$P_{j+1}(f|h) \equiv \frac{\prod_i \alpha_i^{(j+1)g_i(h,f)}}{Z_\alpha(h)^{(j+1)}} \tag{3.29}$$

Continue iterating until convergence or near-convergence.

3.3.2 Improved Iterative Scaling

[20] and [4] give an optimized version of this algorithm called "Improved Iterative Scaling" which is somewhat more complicated and somewhat faster. The off-the-shelf toolkit which we used [43] made use of this optimized algorithm. The algorithm is also described in tutorial form in [42].

This algorithm has the same basic steps as the GIS algorithm, but differs in that it tracks the expected value of each feature in separate "buckets" based on the number of other features which co-occurred with it for each $\langle h, f \rangle$ pair and these numbers are then used in the computation of the next estimate of α.

Specifically, we are doing the following:

1. Compute the expectations of all the g_i's under the current estimate of the probability function subdividing the total expectation by the number of features which fired simultaneously with g_i. Namely, for all pairs $\langle h, f \rangle$ in the training corpus, compute

$$c(h, f) = \sum_l g_l(h, f) \tag{3.30}$$

$$K^{(j)}_{i,c(h,f)} += \tilde{P}(h) \cdot P_j(f|h) \cdot g_i(h, f) \tag{3.31}$$

2. Compute the desired update β_i for every feature g_i:

$$max_c \equiv \text{Maximum value of } c(h, f) \text{ achieved in the corpus} \tag{3.32}$$

   Using Newton's method or some equivalent technique, solve for β_i in

$$\sum_{b=1}^{max_c} K^{(j)}_{i,b} (\beta_i)^b = K_i \tag{3.33}$$

3. Update α_i for the next iteration

$$\alpha_i^{(j+1)} = \alpha_i^{(j)} \cdot \beta_i \tag{3.34}$$

4. The next estimate of the probability function P_{j+1} is defined in the same way as in equation 3.29.

5. Continue iterating until convergence or near convergence.

Note that this algorithm is more numerically fragile than Generalized Iterative Scaling because it runs into overflow problems with high values of max_c. This caused us some difficulties when we were experimenting with "compound features" as we discuss in section 6.2.

Chapter 4

System Architecture

Constructing a maximum entropy modeling system such as MENE is a process of trial and error. The addition of a new class of features will often have the unintended result of reducing system accuracy on unseen data. The modeler needs the ability to experiment with different methods of feature selection and the ability to rapidly port the system to new domains. At the same time, since the modeler may need to run many experiments with different combinations of feature classes, it is important that the time required to run an end-to-end test be kept to a reasonable level.

This chapter will describe how MENE sought to satisfy these competing goals of speed and flexibility.

4.1 History, Future, and Feature Classes

MENE consists of a set of C++ modules and associated Perl scripts which form a wrapper around an off-the-shelf "Maximum Entropy Modeling Toolkit" (MEMT) [43] which computes the values of the α parameters of equation 3.3 from a pair of training files created by MENE. MENE's flexibility is due to its object-based treatment of the three essential components of a maximum entropy system: histories, futures, and features [7].

Since we are trying to assign a future to every token in the corpus, we necessarily have a history corresponding to each token (leaving aside the question of history compression discussed in section 4.4.2). History objects in MENE act as containers for an array of pointers to "history views", which are constructed along with the history objects and are themselves compressible. The history view classes each represent a different type of information about the history object. When the features attempt to determine whether or not they fire on a given history, they request an appropriate history view object from the history object and then query

the history view object to determine whether their firing conditions are satisfied. For instance, the lexical feature described in equation 3.15 which fires when the next word is "said" would request the lexical view from the history object, query the lexical view for w_1's vocabulary index and would then compare this index number to the index number of "said".

Note that these history views generally hold information about a limited window around the current token. If the current token is denoted as w_0, then our model only holds information about tokens $w_{-1} \ldots w_1$ for all history views except the lexical ones. For these views, the window is $w_{-2} \ldots w_2$.

Future objects, on the other hand, are trivial in that their only piece of data is an integer indicating which of the 29 members of the future space they represent.

This architecture enables us to write feature objects which closely mirror their mathematical descriptions. Hence, for a feature object g, the key C++ method is "g.firesOn(h,f)" where h and f are history and future objects, respectively. Since every feature also carries with it its own α value, we can easily iterate over histories, futures, and features to compute $p(f|h)$ as shown in equation 3.3. This architecture also allows us to write very abstract code for such high-level tasks as counting the number of times each feature fires on a training corpus or computing $p(f|h)$. We can keep the high-level code fixed regardless of what types of features are or are not in the feature pool.

4.2 Training the System

As mentioned above, we built many different named entity recognizers with our MENE system. These models were differentiated from each other by the features which were available for inclusion in the model (the "feature pool") and a collection of documents hand-marked with the correct named entities as described in section 2.2 (the "training corpus"). These are turned into a probabilistic model as described in this section. The procedure is summarized in table 4.1.

4.2.1 System Set-up

The first thing the modeler must do is to set up the feature pool. Features are defined to the system via a single file. Each line of the file contains a single feature which is listed in a human-readable form with such information as the class of feature which it represents (i.e. lexical feature, dictionary feature, etc.), its α, an index number by which it is identified to the M.E. toolkit, the future on which it activates, and feature-specific information (i.e. a lexical feature might have the information that it activates on the word "Mr", when that word appears in the w_{-1}

1. Define a training corpus, C
2. Tokenize the training corpus
3. Create a file of candidate features, including lexical features derived from the training corpus
4. Precompute information for the history views from the tokenized file
5. Determine the number of times each feature g_i fires on C ($\#g_i$)
6. Remove features with $\#g_i < m$
7. Create an "expectations" file, with the expected value K_i of every feature g_i over the training corpus ($K_i = \frac{\#g_i}{|C|}$)
8. Create an "events" file listing every feature which activates for every pair $\langle h, f \rangle$ for $h \in C$ and all f
9. Compute the M.E. weightings α_i for every g_i using the MEMT toolkit with the expectations and events files as inputs

Figure 4.1: Training algorithm

position). This file is easily created as the concatenation of feature-class-specific subfiles.

The second step is the tokenization of the training corpus. The text passes through a hand-coded Perl script which tokenizes the text by splitting on white space and anything else which we might not want linked to a named entity. For instance we split the possessive "'s" from the base word and split any "." followed by white space from the previous word. Since the core system is only able to tag entities on a token-by-token basis, we tried to be as liberal as possible in our splitting since we hoped that our system could learn to rejoin improperly split entities (such as the fast-food chain "Arby's"), but it would be completely stymied if a token contained some characters which were inside and some which were outside a named entity (i.e. in the phrase "Clinton's presidency", we must split off the "'s" to properly identify "Clinton" as a person).

Following tokenization, we record information about each token in a series of files which are linked to the master token file on a token-by-token basis. These files serve to precompute various pieces of information about each token. This precomputed information is then read by the various history views mentioned above and is accessed by the associated features. For instance, a separate C++ module run at this point does the dictionary look-ups for the dictionary features described in section 5.4. This module produces a file which says whether each word was or was not in each view's dictionary and, in the case of multi-word dictionary hits, says whether the word was at the start, end, etc. of the dictionary entry.

Although it is true that the feature or history-view objects could have computed this information directly from the tokens, this precomputation is often most easily done in a separate Perl or C++ module. This technique also lets us write simpler feature classes, and it prevents multiple recomputations of the same information during training time. Except for the compound features mentioned in section 5.8, all of the feature classes listed in chapter 5 had a corresponding history view and history view file which contained this precomputed data.

4.2.2 Creating the training files

Now that we have tokenized the text and precomputed important information, we are ready to begin training the features. We first step through the training corpus testing how often $g_i(h, f)$ fires ($\#g_i$) for all features g_i, for all $\langle h, f \rangle$ pairs in the training corpus, C. Next we eliminate from the feature pool those features whose activation count is less than m, where we generally used $m = 3$ (although see the caveats in section 6.1). The remaining features are then written out to an "expectations" file with a probability giving their empirical expectation over the training corpus. This file is an input to the MEMT toolkit and the probabilities represent the values K from section 3.1. Note that $K = \frac{\#g_i}{|C|}$.

We next create an "events" file which lists every $\langle h, f \rangle$ pair in the training corpus along with which features fired on each of these training corpus events. In addition, for every history h_m which occurred, we list every future f_n for which one or more features would have fired if $\langle h_m, f_n \rangle$ had occurred. This file is used in the computation of equations 3.30 and 3.31, which are the key calculations of the Improved Iterative Scaling method (IIS) [20] [4] which is used by the MEMT.

4.2.3 The maximum entropy estimator

Note that in both the events file and the expectations file, histories, futures and features are all identified simply as numbers. Hence the toolkit simply sees the input files as representing a generic M.E. problem which has 29 futures, roughly 320,000 histories, and about 22,000 features (this is the approximate size of the system we entered in the MUC-7 competition).

As we mentioned earlier, the MEMT can, for the most part, be viewed as a "black box", but there are a few things to watch out for:

1. The total number of iterations, j, from section 3.3 is a parameter which must be fed to the toolkit. We found that, in general, performance seemed to peak at around $j = 200 \ldots 400$ (and there was a sharp performance boost between 50 and 100), so we cut off training after 200 iterations.

2. The time required to run the MEMT was not too burdensome (roughly 85 minutes for 200 iterations with the model described above) if the system had sufficient real memory. The MUC-7 model required 338 Megabytes of memory.

   These large memory requirements are necessitated by the need to keep the events file in memory as it is repeatedly passed over during each training iteration. On the other hand, since this data is accessed sequentially, the toolkit doesn't completely fall apart when it is accessing virtual memory, but it does slow by more than an order of magnitude.

3. Care has to be taken not to define too many high-frequency features. High frequency features increase the size of the events file (consequently increasing memory requirements). In addition, the IIS algorithm runs into numerical problems when too many features fire on the same $\langle h, f \rangle$ pair (because the exponent in equation 3.33 is tied to the number of simultaneously firing features). We ran into trouble with the MEMT when more than about 50 features fired simultaneously in this way. For instance, one does not want to define a feature which activates when a word is *not* on a list of states of the U.S. since this would fire almost everywhere. It is far better to have a feature which activates when a word is an American state.

After the estimator has completed its run, we have a list of features G which we want included in the model and for every $g_i \in G$, we have an M.E. weighting α_i. The model is trained.

4.3 Decoding

After having trained the features of an M.E. model and assigned the proper weight (α value) to each of the features, decoding (i.e. "marking up") a new piece of text is a fairly simple process:

1. Tokenization (same as in section 4.2.1)
2. Precomputation of the history views (same as section 4.2.1).
3. For each article of the text
   (a) For each history, h, (i.e. a token) of the text, check each feature g to see whether there is a future, f, such that $g(h, f)$ holds. Combine the α values of the firing features according to equation 3.3. This will give us a conditional probability for each of the 29 futures for each token in the article.
   (b) Run a Viterbi search to find the highest probability legal path through the lattice of conditional probabilities, thereby producing a sequence of tags from our future space.
4. Translate these token-by-token taggings (which might be a sequence like [person_start, person_continue, person_end, other, other, location_unique]) into SGML markings of the original text as specified by the MUC-7 guidelines. This is a simple deterministic process.

The Viterbi search is necessary because simply taking the highest-probability future assigned to each token would result in incompatible assignments. For instance, an assignment of [person_start, location_end] to two consecutive tokens would be invalid. The Viterbi search finds the highest probability path in which there are no two tokens in which the second one cannot follow the first, as defined by a table of all such invalid transitions (a similar approach to [48]).

4.4 Technical Information

Making the system run acceptably fast was crucial to our ability to run a sufficient number of experiments to test the model. In this section we will look at some technical points which contributed to MENE's performance.

4.4.1 Hardware and software platforms

MENE was implemented using the egcs compiler [52], a free software project aimed at (among other things) implementing the ANSI/ISO C++ standard [1] [51]. The use of the egcs implementation of ANSI/ISO C++ facilitated MENE's development primarily through its support of the following:

- Built-in strings
- Support for data structures such as maps, multi-maps, sets and bit vectors, which were used in our features and in the optimizations described below.
- Built-in algorithms such as sort routines
- Excellent support for templates. Most of the features listed in chapter 5 were instantiations of a single generalized template. This greatly speeded code development.
- Cross-platform support–the system ported smoothly between the Solaris and Linux operating systems

While it certainly would have been possible for the author to have implemented the above-listed data structures and algorithms himself or to have used a heterogeneous collection of publicly available libraries, the availability of high-quality, consistent implementations in the C++ Standard Library greatly speeded system development.

More generally, the use of C++ was very useful to the project. Features and history views were both implemented as abstract base classes which were specialized for the various types of features and history views which we used in the system. This structure allowed us to easily add and subtract new features and history views from the system. While there are certainly many languages which offer inheritance and objects, unlike most of these languages, C++ is very fast, which was crucial given the large data sizes we were working with.

All performance statistics quoted in this dissertation are for a 143 MHZ Sun Ultra-1 machine with 512 megabytes of RAM running Solaris 2.5 or 2.6.

4.4.2 History compression

The calculations of equation 3.27 can be speeded up if we collapse functionally identical tokens into a single history, h, while keeping track of the count of h so that we can accurately compute $\tilde{P}(h)$. This is easily done using the above-mentioned C++ Standard Library maps by implementing a definition of equality for each history view so that the map "knows" when to collapse two histories

into one. For instance, since the lexical history views discussed in section 5.2 hold information about words relative to a fixed vocabulary, words not in the vocabulary are all mapped to the "unknown token". If we assume that both "Timbuktu" and "Kalamazoo" are not in our vocabulary, then the equality function for the lexical history view would see the phrases "flying to Timbuktu from Chicago" and "flying to Kalamazoo from Chicago" as being equal. In order to compress two histories, all of the corresponding history views for each history must be compressible, so if some other history view made a distinction between "Timbuktu" and "Kalamazoo", we would not be able to collapse these two histories.

Note that this compression strategy requires us to hold the entire training corpus in memory so that we can find compressible histories. This memory expense is mitigated by the fact that we can compress equivalent history views even in cases when the histories are not compressible. Furthermore, the memory requirements of the MENE training routines are overshadowed by the requirements of the MEMT (on the system entered in MUC-7, the program which created the "events" file required 114 meg to process the 2.3 meg training corpus while the MEMT required 338 meg). Finally, note that history compression only has a strong theoretical benefit when creating the "events" file which is used by the maximum entropy estimator, since the MEMT passes over every history 200 times. Specifically, when decoding new text, one can simply create and discard history objects one-by-one.

Unfortunately, these compressible histories were relatively rare in our application, so we were only able to reduce the number of histories by 2.7% with this method. While this probably wasn't a sufficient level of compression to make the technique worthwhile for MENE, we think that it could be very useful if the system were ported to an application in which there was a greater degree of history duplication.

4.4.3 Feature caching and future retrieval

Feature caching was much more successful for us. In the computation of the conditional probability in equation 3.3, the counting of feature firings or the creation of the events and expectations files mentioned in section 4.2.2, a naive implementation would simply iterate across every feature, g_i for every $\langle h, f \rangle$ pair testing whether g_i.firesOn(h,f) holds.

This approach would be particularly disastrous for computing conditional probabilities at decoding time since for every feature we would have to check every $f \in F$ for every h in the training corpus. As a first optimization, we noted that all of the features in our model are separable (as discussed in the appendix, this means that $g(h, f) = g'(h) \wedge g''(f)$, i.e. that the history and future tests are separate). Thus we can first test whether $g'(h)$ holds and then query the feature g to get the list of $f \in F|g''(f)$. Although one could imagine many features which would fire

on more than one future, for all of the features used in MENE, this list had only one member. This technique reduced the time complexity of a file traversal from $O(|\text{Histories}| \cdot |\text{Futures}| \cdot |\text{Features}|)$ to $O(|\text{Histories}| \cdot |\text{Features}|)$.

The next step was to eliminate the time dependency on the number of features in the model. The vast majority of features in our model (93.2% of the MUC-7 model) were lexical features. As described in section 5.2, the history side of a lexical feature ($g'(h)$ in the above equation), is characterized by a vocabulary index and an offset from the current token (i.e. $w_{-1} = index\_of(\text{“mr”})$). Let $M = index\_of(\text{mr})$, then we can characterize this feature as $\langle -1, M \rangle$. In a MENE parameters file, we designate lexical features as “cacheable” (the external system features described in section 5.5 constituted 3.3% of the MUC-7 feature pool and were also designated as cacheable in a manner similar to the lexical features).

As the features are loaded from a file, non-cacheable features are stored in a simple array and are all checked for each history. Cacheable features, though, are stored in a multi-map (a map where a single key can return multiple items) keyed by the integer pair described above. When decoding, we retrieve the cached features for each of the tokens in the 5-word-window which we maintain around the current word (from $w_{-2} \ldots w_2$). We then test every feature, g, thus retrieved to see if $g'(h)$ holds. This last step is necessary because some features have additional restrictions besides those listed in their key. Lexical features, for instance, are looking for a certain word at a certain offset, but they also have the restriction that they (like every other feature except for the global reference resolution features discussed in section 5.6) may not look at words across document boundaries. I.e. if w_{-2} were in a different document from w_0, then a lexical feature looking for a particular token in w_{-2} would not fire under any circumstances. When $g'(h)$ holds, we query g for its futures as described above.

4.4.4 System time requirements

A complete training run for the feature pool and training corpus used in the MUC-7 evaluation (roughly 22,000 features, and a 350-article (321,000 word) corpus) took approximately 2 hours 40 minutes. We tagged the 100-article (72,000 word) MUC-7 test corpus in 23 minutes, a speed of about 1 megabyte of text per hour.

We found the training time to be sufficiently fast for system development. The speed of tagging was also ample for research purposes, but it would likely be deemed inadequate for a commercial implementation. BBN's HMM-based system ran at 6 MB/hr [5] on comparable hardware and IsoQuest achieved 382 MB/hr on a machine roughly twice as fast as ours [29]. In a production-oriented system, we think that MENE's speed could be greatly enhanced by eliminating the precomputation of the history views and moving this logic into the histories and features. The precomputation gives greater efficiency at training time because we traverse

the same text 3 - 4 times in a single training run and we do multiple training runs on the same piece of text, but there is no time benefit at tagging time. In addition, we are confident that if the system were rewritten with more of an eye for speed (and perhaps with some loss of flexibility), one could achieve enormous speed-ups.

Chapter 5

Types of Features

In this chapter we will look at some of the different feature types which we have experimented with in the MENE system.

5.1 Binary

While all of MENE's features have binary-valued output, the "binary" features are features whose associated history-view can be considered to be either on or off for a given token. The binary history view was thus easily implemented as a bit-vector. Equation 3.2 gives an example of a binary feature. MENE's binary features are listed in table 5.1.

	Description	Example Text	Intuition
1	2-digit Number	99, 84	Two-digit-year
2	4-digit Number	1999, 1776	Four-digit-year
3	Only digits	5, 1999, 242	Misc. numbers
4	Mixed letters and digits	F14	Product code
5	Number with comma	2,000	Monetary amount
6	Number with period	42.56	Monetary amount, percentage
7	A valid number	-3.15, .12, 3,000.43	All numbers
8	All-caps	BBN	Organization
9	Initial Cap	Bill, Intel	Capitalized word
10	Uncapitalized word	can, won't	Probably not an entity
11	Internal Capitalization	ValuJet	Corporate name

Table 5.1: Binary features used in MENE

These binary history-views used by MENE's binary features are very similar

to those used in BBN's Identifinder system discussed in chapter 2.4 with three exceptions:

- Features 3 (only digits) and 11 (internal capitalization) were not found in the BBN system.
- Identifinder used a feature for "significant" (i.e. non-sentence-beginning) capitalization. We didn't include this, believing that MENE could make these judgments from the surrounding lexical content.
- Identifinder's features were non-overlapping. I.e. the all-cap feature took precedence over the initial-cap feature. Given two features, a and b, when the $\langle$history,future$\rangle$ space on which feature b activates must be a subset of the space for feature a, it can be shown that the M.E. model will yield the same results whether a and b are included as features or if $(a - b)$ and b are features (see appendix A for a proof). Consequently, MENE allows all features to fire in overlapping cases. For instance, in MENE the initial cap features activate on the histories "Clinton", "IBM", and "ValuJet" while under a hierarchical, non-overlapping scheme like Identifinder's, the feature would only be active on "Clinton" because the "AllCap" feature would take precedence on "IBM" and an "Internal Capitalization" feature would take precedence on "ValuJet".

5.2 Lexical

To create a lexical history view, the tokens at $w_{-2} \ldots w_2$ are compared with a vocabulary and their vocabulary indices are recorded. Words not found in the vocabulary are assigned a distinguished "Unknown" index. The following is an example:

$$g(h, f) = \left\{ \begin{array}{lcl} 1 & : & \text{if Lexical-History-View}(token_{-1}(\text{h})) = \text{"Mr" and} \\ & & f = \text{person\_unique} \\ 0 & : & \text{else} \end{array} \right\}$$

- Correctly predicts: Mr **Jones**

A more subtle feature picked up by MENE: preceding word is "to" and future is "location_unique". Given the domain of the MUC-7 training data (aviation disasters), "to" is a weak indicator, but a real one. This is an example of a feature which MENE can make use of but which the constructor of a hand-coded system would probably regard as too risky to incorporate. This feature, in conjunction with other weak features, can allow MENE to pick up names that other systems might miss.

An important point to emphasize is that these features were not hand-selected. We build the list of lexical features to be included in a model as follows.

1. Define a vocabulary, V, as "all words appearing in the training corpus with a count of three or more".

2. Include in the feature pool features with every possible feature which can be characterized by the 3-tuple $\langle w, f, n \rangle$ where $w \in \{V \cup \text{"Unknown word"}\}$, f is any of the 29 futures, and $n \in -2 \ldots 2$.

3. As with all other feature classes, select features from the feature pool which occurred at least three times (see chapter 6.1 for some minor caveats).

The attractive thing about this algorithm is that it does not involve any linguistic intuition, so these lexical features are completely portable to new domains and even new languages. This is very encouraging since, as discussed in section 7.4, the system can attain a reasonable level of performance using these features alone.

5.3 Section

The New York Times articles which constituted the MUC-7 test and training corpora were composed of six distinct sections including "Date", "Preamble", and "Text". Section features activate according to which of these sections the current token is in. Example feature:

$$g(h, f) = \left\{ \begin{array}{lcl} 1 & : & \text{if Section-View}(token_0(\text{h})) = \text{"Preamble" and } f \\ & & = \text{person\_unique} \\ 0 & : & \text{else} \end{array} \right\}$$

Activation example: **CLINTON** WARNS **HUSSEIN** ABOUT IRAQI DEFIANCE. Note that, assuming that this headline is in the preamble, the above feature will fire on *all* of these words. Of course, this feature's prediction will only be correct on "CLINTON" and "HUSSEIN".

Section features serve the dual purpose of differentiating the prior probability of the different futures occurring in each section, and, since training and test corpora are partitioned by these sections, we are also implicitly establishing the prior probability of the futures over the entire corpus (as discussed in appendix A). For instance, in NYU's evaluation system, the α value assigned to the feature which predicts "other" given a current section of "main body of text" is 7.9 times stronger than the feature which predicts "person_unique" in the same section. Thus the system predicts "other" by default. On the other hand, in the preamble (which contains headline, author, etc. information), the weight of the feature predicting

"other" is much weaker relative to most of the "non-other" futures. It is only about 2.6 times as strong as "organization_start" and "organization_end", for instance.

As discussed in section 7.4, subsequent to the MUC-7 evaluation, we found that deleting all section features from the model had no impact on system performance. This was a surprising result to us, so we can only speculate as to the cause. Our intuition, though, is that the system had enough other very-high-frequency features (such as the binary features which fired on capitalized and non-capitalized words) that the "job" of establishing features' prior probabilities was shifted from the section features onto these other ones. Note also that we have at least one lexical feature firing on every token, since every token is either in our vocabulary or is an unknown word. Thus we do not really have many situations where the system has to rely on section features alone.

5.4 Dictionary

Multi-word dictionaries are a key element of MENE. Each entry in a MENE dictionary consists of a term which is one or more tokens long. Dictionaries can be case-sensitive or not on a dictionary-by-dictionary basis. A pre-processing step summarizes the information in the dictionary on a token-by-token basis by assigning to every token one of the following five tags for each dictionary: start, continue, end, unique, other. I.e. if "British Airways" was in our dictionary, a dictionary feature would see the phrase "on British Airways Flight 962" as "other, start, end, other, other". Table 5.2 lists the dictionaries used by MENE in the MUC-7 evaluation. Below is a specific example of a dictionary feature:

$$g(h,f) = \left\{ \begin{array}{lcl} 1 & : & \text{if First-Name-Dictionary-View}(token_0(\text{h})) = \\ & & \text{"unique" and } f = \text{person\_start} \\ 0 & : & \text{else} \end{array} \right\}$$

- Example: **Richard** M. Nixon–assuming that "Richard" is in the first name dictionary.

Note that, similar to the case of overlapping binary features, we don't have to worry about words appearing in the dictionary which are commonly used in another sense. I.e. we can leave dangerous-looking names like "April" in the first-name dictionary because whenever the first-name dictionary feature fires on April, a lexical feature for April will also fire and, assuming that the use of April as "date" exceeded the use of April as person_start or person_unique, we can expect that the lexical feature will have a high enough α value to outweigh the first-name-dictionary feature.

Under certain circumstances, we can rigorously prove that this will happen by using the result from appendix A. First let us define these three features:

Dictionary	Number of Entries	Data Source	Examples
First names	1245	www.babyname.com	John, Julie, April
Corporate names	10,300	www.marketguide.com	Exxon Corporation Motorola, Inc.
Corporate names without suffixes	10,300	"corporate names" processed through a Perl script	Exxon; Motorola
Colleges and universities	1225	www.utexas.edu/ world/univ/alpha/	New York University Oberlin College
Corporate Suffixes	244	Tipster resource	Inc.; Incorporated
Dates and times	51	Hand Entered	Sunday, April, EST
2-letter state abbreviations	50	www.usps.gov	NY, CA
World Regions	14	www.yahoo.com	Africa, Caribbean Pacific Rim

Table 5.2: Dictionaries used in MENE

1. w_0 is "April" predicting person_start
2. w_0 present in first-name-dictionary predicting person_start
3. w_0 present in first-name-dictionary where the dictionary has been edited to exclude "April" predicting person_start

Clearly the first and third features above partition the space defined by the second feature. Thus by the proof, including any two of these three features in the model will give the same effect as using all three.

On the other hand, this proof assumes that we have a feature in which April predicts person_start. This is unlikely given our feature-selection algorithm since April is unlikely to appear three times as a person's first name in any reasonable sized training corpus.

Let us, however, consider this more likely case. Consider the following features:

1. w_0 is "April" predicting date_start
2. w_0 is "April" predicting date_continue
3. w_0 is "April" predicting date_end
4. w_0 is "April" predicting date_unique

5. w_0 present in first-name-dictionary predicting person_start

6. w_0 present in first-name-dictionary where the dictionary has been edited to exclude "April" predicting person_start

Let's suppose that April *always* appeared as a date in the training corpus. Then we could assume that a model built from this corpus contained an imaginary lexical feature predicting person_start given April because it would have its constraints satisfied. This is because the model would be assigning $P_M(\text{person\_start}|\text{April}) \approx 0$ (where P_M is the model's probability) whether or not this feature was included because features 1-4 would force $\sum_x P_M(\text{date\_}x|\text{April}) \approx 1$. Hence, by the first proof of this section, we wouldn't care whether or not the dictionary contained April.

However, this discussion raises the larger point that dictionary entries which have corresponding lexical features for a given future have no impact on the dictionary's prediction of that future. This is clear from the appendix proof because as a generalization of the earlier proof, if we define:

1. t, a subset of the vocabulary. S is a set of lexical features s_i with s_i predicting future f_k with history condition t_i.

2. e is a dictionary feature over the dictionary d predicting f_k

3. e' is a dictionary feature over the dictionary $d - t$ predicting f_k

then we can clearly see that S and e' partition e, so including the features S and e' gives an equivalent result to including S and e. Consequently, the only entries in our dictionary which are relevant to system performance are those which either don't appear in the vocabulary (including multi-word entries) or those which do occur but co-occur with a given future less than three times, thus falling below the cut-off for inclusion in the model as a feature. On the other hand, dictionary entries which don't occur in the training corpus at all will be irrelevant for training purposes. Therefore, for single-word entries, the weighting assigned to the dictionary features will be solely decided by those entries which co-occur just once or twice with a particular future.

Having considered the cases where "April" appears often enough as a person to have a person_start lexical feature and the case where April doesn't appear at all as a person, we are now ready to consider the case where April appears once or twice as a person, forcing $\sum_x(P_M(\text{date\_}x|\text{April}))$ to average less than 1 over the training corpus. In this case, the choice of features 5 or 6 above would affect the model's behavior somewhat, but it seems preferable to leave the word in the person dictionary (i.e. choose feature 5) for the following reasons:

- The choice of 5 or 6 will not affect the average $P_M(\text{date\_}x|\text{April})$ over the training corpus because this must equal the empirical average $\tilde{P}(\text{date\_}x|\text{April})$ because of the presence of features 1 through 4 (by equation 3.13).

- Assuming that the number of times April is a date greatly outweighs the number of times April is a person, we can assume that the average $P_M(\text{person\_start}|\text{April})$ over the training corpus will be small.

- The choice of feature 5 or 6 *does* affect the probability distribution between person_start and the other non-date futures for April. However, it makes sense that we should use the unedited dictionary so that person_start will have a higher probability than these other futures in this situation.

- As we have argued, it is precisely those words like April which occur rarely as person names which our dictionary feature is modeling. Hence, we should be disinclined to selectively delete words in this category from our list.

These arguments were supported by the results of our test runs: no instance of April was tagged as a person name, including one case, "The death of Ron Brown in April in a similar plane crash ..." which could be thought of as somewhat tricky because the month was not followed by a specific date.

These results tell us that the behavior of a new dictionary added to the model might be counter-intuitive, particularly for dictionaries of single-word items. We found this during our experimentation with adding dictionaries to the system. There were a number of dictionaries which we rejected for inclusion in the MUC-7 system because after training we found that system performance on test data decreased relative to a baseline system. Some of these are summarized in table 5.3.

Dictionary	Number of Entries	Data Source	Examples
Location identifiers	206	www.usps.gov	Avenue, Harbor, Meadows
Worldwide cities	1358	www.cnn.com	London, New Delhi
U.S. states	51	www.yahoo.com	Alabama, New York, Washington, D.C.
World airlines	256	web search	Aeroflot, Air France, Royal Jordanian
Aviation org's	198	web search	Civil Air Patrol, FAA

Table 5.3: Some dictionaries rejected for MENE

5.5 External System

NYU's official entry in the MUC-7 evaluation was a MENE system which took in the output of an enhanced version of the more traditional, hand-coded "Proteus" named-entity tagger discussed in section 2.1. In addition, subsequent to the evaluation, the University of Manitoba [30] and IsoQuest, Inc. [29] shared with us the outputs of their systems on our training corpora as well as on various test corpora. The output sent to us was the standard MUC-7 output, so our collaborators didn't have to do any special processing for us. These systems were incorporated into MENE as simply three more history views by the following 2 step process:

1. Each system's output is tokenized by MENE's tokenizer and cross-system tokenization discrepancies are resolved.

2. The tag assigned to each token by each system is noted. This tag will be one of the 29 tags mentioned above (i.e. person_start, location_continue, etc.)

The result of all this is that the "futures" produced by the three external systems become three "external system history views" for MENE. The following gives a function q which is the general form of this feature class along with a specific example, g:

$$
\begin{aligned}
j &\in -1 \ldots 1 \\
x &\in \{\text{Proteus, Manitoba, IsoQuest}\} \\
y, z &\in F = \{\text{The space of 29 futures}\} \\
q(h, f) &= \left\{ \begin{array}{lcl} 1 & : & \text{if } x\text{-System-View}(token_j(h)) = y \text{ and } f = z \\ 0 & : & \text{else} \end{array} \right\} \\
g(h, f) &= \left\{ \begin{array}{lcl} 1 & : & \text{if Proteus-System-View}(token_0(\text{h})) = \text{person\_start and } f = \text{person\_start} \\ 0 & : & \text{else} \end{array} \right\}
\end{aligned}
$$

- Example: **Richard** M. Nixon, in a case where Proteus has correctly tagged "Richard".

It is important to note that MENE has features which predict a different future than the future predicted by the external system. This can be seen as the process by which MENE learns the errors which the external system is likely to make. An example of this is that on the evaluation system the feature which predicted person_unique given a tag of person_unique by Proteus had only a 76% higher weight than the feature which predicted person_start given person_unique. In other words, Proteus had a tendency to chop off multi-word names at the first word. MENE learned this and made it easy to override Proteus in this way. In

fact, an analysis of the differences between the Proteus output and the MENE + Proteus output [49] turned up a significant number of instances in which MENE extended or contracted name boundaries in this way.

We found a similar example in the MENE system which took the Manitoba output as an input. The MUC-7 scorer did not penalize systems for putting an extraneous "." on the end of a tagged string. I.e. for the abbreviation of Minnesota, it would count both [⟨ENAMEX TYPE="LOCATION"⟩Minn.⟨/ENAMEX⟩] and [⟨ENAMEX TYPE="LOCATION"⟩Minn⟨/ENAMEX⟩.] as correct. This was true even if the "." clearly wasn't a part of the named entity, as in [⟨ENAMEX TYPE="LOCATION"⟩Minnesota.⟨/ENAMEX⟩]. The Manitoba system exploited this loophole in the scoring algorithm by always including a "." along with a contiguous named entity (so it would have included the "." in both of the above cases). The MENE-Manitoba system, however, was trained off of the hand-tagged data in which the human taggers had made their own judgement about what was the correct entity boundary, (in particular, the "." was part of the entity on "Minn.") and had no knowledge that the scorer was lenient in this regard. This system thus "learned" in training its features that the Manitoba system had a tendency to add an extraneous period onto the end of entities. This could be seen in the weighting of features, where, for instance, the feature predicting person_end for w_0 given a Manitoba tagging of person_end to w_1 was weighted 19% higher than the more intuitive feature predicting person_end for w_0 given that Manitoba made that prediction on w_0. Another example can be seen by comparing the formal run results, where the MENE-Manitoba system only "erroneously" appended the "." onto four named entities while its input from the Manitoba system contained roughly 450 entities with this "error".

Thus we can see that given proper training data, MENE can pinpoint and selectively correct the weaknesses of a handcoded system. This is further demonstrated in the results in chapter 7, where we show that the Proteus, Manitoba, and IsoQuest systems were all improved by this hybridization with MENE.

5.6 Reference Resolution

Analysis of the reasons for the higher performance in the MUC-7 formal run of the MENE-Proteus system vs the MENE-only system [49] determined that much of the improvement from adding the Proteus system was due to Proteus' use of long-distance reference resolution. Reference resolution involves finding words which seem to co-refer to the same entity. If one of them is tagged as a particular name class, then it is likely that all such words should also be tagged with that name class. As an example, we would expect such a reference resolution feature to help us out in an article in which "Andrew Borthwick" was followed by a subsequent

reference to "Borthwick". The first reference would be fairly easy for MENE to pick up since it involves a common first-name. The second instance would be more difficult. Since "Borthwick" is a rare word, it would not be included in our vocabulary and thus would not trigger a lexical feature strongly predicting "person-unique" as a more common name like "Smith" would. Consequently, in the absence of strong contextual evidence (like being preceded by "Mr." or followed by "said"), MENE would probably mistag it. This problem would be avoided if there were a feature predicting "person-unique" whose history condition was that the current word was tagged as "person" somewhere else in the article.

Our reference resolution features were based on a method described in Edinburgh/LTG's MUC-7 named entity paper [35]. LTG recognized a name as being an alias for another name if the tokens in the new name were an ordered subset of the already-recognized name. Thus, if "George Herbert Walker Bush" were recognized as a name, our reference resolution feature would fire on "George Herbert", "George Bush", "George", and "George Herbert Bush", but would not fire on "Herbert George". In addition, we have a "stoplist" of certain common single-token words which should not be tagged by the feature, including "in", "the", "and", etc. Acronyms are generated by the same method. For instance, given the above-cited example our acronym-resolution feature would fire on "GHWB", "GB", "GH", etc. Single-letter acronyms are not allowed, so the acronym feature wouldn't fire on "B". One final technical detail is that we had different features firing depending on whether the reference was in the current article or in a different article in the corpus and depending on the number of times the sequence occurred elsewhere (specifically 1, 2, or 3+). Hence, the general form of the feature class was the following:

$$
\begin{aligned}
n &\in \{1, 2, 3+\} \\
u &\in \{\text{“start”, “middle”, “end”, “only member of”}\} \\
v &\in \{\text{“words”, “first letters of words”}\} \\
x &\in \{\text{“organization”, “person”, “location”}\} \\
y &\in \{\text{“current document”, “current corpus”}\} \\
z &\in F \\
g(h, f) &= \left\{ \begin{array}{lcl} & & \text{if } w_0 \text{ is the } u \text{ of a sequence of } v \text{ which are an} \\ 1 & : & \text{ordered subset of a sequence which was tagged } n \\ & & \text{times as a } x \text{ elsewhere in the } y \text{ and } f = z \\ 0 & : & \text{else} \end{array} \right\}
\end{aligned}
$$

Note that the inclusion of these features requires that MENE work in two passes and incorporate two separately trained models. When decoding, the first pass uses a model trained with all of MENE's features except for the long-range ones. The second pass uses a model containing all the same features plus the long-range

reference resolution/acronym features described above. These new features both train and decode by looking at MENE's first pass output. Consequently, training MENE with these new features becomes a three-step process:

1. Train a model incorporating all features other than the long-range ones as described in chapter 4.

2. Run the resulting model against the training data used to train it in step 1.

3. Train a second model with the same corpus and features as the first step plus the long-range features. The long-range features take as their input the output of step 2.

One could have trained the model by having the long-range features look at the answer keys during training, but the advantage of training a model this way is that it trains on imperfect data. The model "learns" that the output of the first-pass model is sometimes in error, and lessens the α values assigned to the long-range features accordingly. However, there is still a significant problem with this training method since the model's performance on training data is significantly higher than it is on unseen data, as one would expect. Consequently, the reference resolution features will be somewhat overweighted.

In an attempt to compensate for this overweighting, we divided the corpus into five segments of 20% each. We then trained five models on 80% of the data and tested on the other 20%, rotating the data appropriately. Finally, we concatenated the 20% tests and trained the model with reference-resolution features on this corpus. As shown in table 7.5, we found that this method gave us our best results in both the MENE-only and MENE-Proteus categories. We also saw that this data rotation improved the model's performance by a small but significant margin over its performance without data rotation.

5.7 Consistency

The system as it currently stands and as it was entered in the MUC-7 evaluation relies on a process external to the maximum entropy model to enforce the requirement that MENE output a consistent sequence of futures (i.e. that it did not output something like the sequence ["person_start" "location_end"]). At the beginning of our development process we experimented with including in the model a single "consistency" feature which would test whether the future for w_0 was consistent with the future assigned to w_{-1}. For instance it would activate if w_{-1} was tagged as "person_start" and the future (the tag for w_0) was "person_continue". This same feature activated when w_{-1} was "location_continue" and the future was

"location_end". Since this feature was looking at the future for the previous token as its "history", it, unlike all other features, had to operate on the edges of a lattice connecting each possible future for w_0 to each of the 29 possible futures which could have been assigned to w_{-1}. This lattice of edges is then navigated by the Viterbi search to find the highest probability valid sequence as described in section 4.3. Note that invalid edges are given a probability of zero by the external process regardless of whether or not we were using the consistency feature.

During decoding, this feature has a relatively small impact because our decoding algorithm "artificially" sets the probability of any edge representing an illegal transition to zero and the consistency feature boosts the probability of all consistent futures for a given prior future by the same amount. Therefore, our thinking was that the feature might have its greatest impact by influencing the training of other features. For instance, a feature predicting "person_end" given that w_{-1} = "Andrew" might be given a lesser weight when the model included a consistency feature since "Andrew" would likely be tagged as "person_start" in the training corpus and the consistency feature would be firing only on "person_continue" and "person_end" in this context. Our theory was that these features would be deemphasized while features making predictions which are not easily captured by the consistency feature (for instance the feature predicting "person_start" when w_0 is "Andrew") would be emphasized.

We discovered empirically that the consistency feature was not useful. Deleting it from the model had no impact on system accuracy. On reflection we felt that this was not such a surprising result. This feature was trained with the answer key as its input, which would tend to give it an excessively high weighting since the answer key is always correct. At run time it will be activating on many different hypotheses for the tagging of w_{-1}, some of which will be correct and some incorrect. An inconsistency between the training and testing characteristics of a feature generally leads to reduced performance.

This feature also had the drawback of significantly expanding the number of feature activations during training and testing, which slowed the system and increased memory and disk usage. Consequently we removed it from our system and didn't use it during the MUC-7 evaluation.

We haven't attempted using more sophisticated consistency features in the system, such as having a different consistency feature for each legal future - future transition. An example of this would be a feature predicting "person_continue" given that the previous token was tagged as "person_start". Such a system would be similar to Ratnaparkhi's part of speech tagger [40] in which the tagger looked at the POS tags it assigned to w_{-1} and w_{-2} when making its decision on w_0 and then found the best overall sequence of tags by using a beam search. However, given the above arguments and the fact that, contrary to the case with POS tagging, we are able to absolutely exclude most transitions, we are doubtful that this sort of

feature class would be useful.

5.8 Compound

The MENE system entered in MUC-7 had no direct ability to learn compound features or "patterns"–the "history" side of a lexical feature activated based on only a single word, for instance. A sort of pattern-like ability came into the system from multiple features firing at once. I.e. to predict that "York" in the name "New York" is the end of a location, we had two features firing: one predicts location_end when $token_{-1}$ is "new". The other predicts location_end when $token_0$ is "york".

Nevertheless, it is possible for a compound feature to behave differently from two simultaneously firing "atomic" features. As an example, the MUC-7 system integrated compound features into the model in an *ad hoc* manner for the external system features, where we constructed features which essentially queried the external system history and the section history simultaneously to determine whether they fired. I.e. a particular feature might fire if Proteus predicts person_start, the current section is "main body of text", and the future is "person_start". This allows MENE to assign a lower α to a Proteus prediction in the preamble vs. a prediction in the main body of text. Proteus, like many hand-coded systems, is more accurate in the main body of the text than in headline-type material. We found that this compound feature gave the system slightly higher performance than we got when we just used section features and external system features separately.

Encouraged by this result, we implemented a fully general compound feature (i.e. feature A fires if features B and C both fire). This feature was easily implemented in about a day's work as simply a feature class which incorporated an array of two or more arbitrary features as member data. Note that we didn't have to write a new "history view" for this–the constituent features simply queried their own history views. The following is the general form of our compound features:

$$g_{ab}(h, f) = g_a(h, f) \wedge g_b(h, f)$$

Our intuition was that we would see a performance gain from adding these features. Some of the useful feature combinations which we expected to see were:

- w_0 capitalized and w_0 in main body of text predicts person_unique (or any future besides "other"). Capitalization is more significant in the main body of the text than in the preamble (i.e. in headlines).

- Previous word is "." and current word is capitalized, predicting "other". This is a crude way of saying that capitalization is not significant at the start of a sentence.

- Short phrases. I.e. w_{-2} = "flying" and w_{-1} = "to" predicts "location_unique".

Incorporating compound features into the model involved significant questions of identifying a feature pool of co-occurring features, watching out for features whose effective frequency was too low for their statistics to be meaningful (since a compound feature defines a subset of the space defined by its atomic features, we run into the issues discussed in appendix A), and then selecting a small enough subset of this pool so that the system wouldn't become overloaded. These issues are discussed in section 6.2. As we state in that section, we have thus far been unable to elicit any consistent performance gains from these features.

Chapter 6

Feature Selection

This chapter will look in detail at our method for selecting the features which we included in MENE's models. Section 6.1 will look at the methods which we used to build the model entered in the MUC-7 evaluation. Section 6.2 will look at some techniques which we used in an attempt to incorporate compound features into the model.

6.1 Selecting by Count

Our basic feature selection method is very simple. First we put all possible features from the classes we want included in our model into a "feature pool". For instance, if we want lexical features in our model which activate on a range of $w_{-2} \ldots w_2$, our vocabulary has a size of V, and we have 29 futures, we will add $(5 \cdot (V+1) \cdot 29)$ lexical features to the pool. The $V+1$ term comes from the fact that we include all words in the vocabulary plus the unknown word. From this pool, we then select all features which fire at least three times on the training corpus.

Note that this algorithm is entirely free of human intervention. Once the modeler has selected the classes of features, MENE will both select all the relevant features and train the features to have the proper weightings.

We deviate from this basic algorithm in three ways:

1. We exclude features which activate on some sort of "default" value of a history view. Many history views have some sort of default value which they display for the vast majority of tokens. For instance, a first-name-dictionary history view would say that the current token is not in the dictionary in over 99% of the cases. Rather than adding features which activate both when the token in question is and when it is not in a dictionary, we only include features which activate when the token **is** in the dictionary. A feature which

activated when a token was not found in the dictionary, while theoretically not harmful, would have practical disadvantages.

   (a) If for a given future, f_k, we included in the model both the "is in dictionary" and "not in dictionary" features, then these features would partition the space defined by a feature which always predicted f_k. By appendix A, our model would be functionally equivalent if any two of these three features were included. Although it is true that we don't currently have such a "future-only" class of features in the model, by including the section features described in section 5.3, we have at least approximately partitioned the space which would have been defined by such a feature. Consequently, our model is roughly equivalent to a model which contained such a feature, hence by the proof it is not necessary to build a model which contained, for a given future, the "is in" and "is not in" dictionary features.

   (b) Since this feature would fire on nearly every token, it would slow down run-time performance.

   (c) While maximum entropy models are designed to handle feature overlap, a very high degree of overlap requires more iterations of the maximum entropy estimation routine and can lead to numerical difficulties [43].

2. Features which predict the future "other" have to fire six times to be included in the model rather than three. Experiments showed that doing this had no impact on performance and reduced the size of the model by about 20%.

3. As another way of reducing the model size, lexical features which activate on w_{-2} and w_2 are excluded if they predict "other". Like the previous heuristic, this is based on the idea that features predicting named entities are more useful than features predicting "other".

6.2 Techniques Used in Compound Feature Selection

While our method of selecting features worked well for the "atomic" (i.e. non-compound) features in our model, we needed a more sophisticated method for selecting compound features due to the enormous number of potential compound features. Although at this point in our research we have not been able to get a measurable benefit from the addition of compound features to the model, we feel that the techniques which we used in attempting to add them are interesting in their own right. These techniques could be useful in any case in which one is

seeking to choose the best features to add to a preexisting model from a vast pool of potential features.

Selecting these features was a multi-stage process which can be summarized as follows (recall in the following that $\#g$ is the number of times that feature g fires on the training corpus):

As a first cut, we define a pool of compound features to be every pair of features which met the following two conditions.

1. If we define $f_{ab}(h, f) = f_a(h, f) \wedge f_b(h, f)$, then we require that $\#f_{ab} > k$, which is the same as selecting features by count as we described in section 6.1.

2. We also require that $\#f_a - \#f_{ab} > k$ and $\#f_b - \#f_{ab} > k$.

We can see that the second condition is necessary if we consider the following: define $f_{a\overline{b}} = f_a \wedge \overline{f_b}$. Then it is clear that f_{ab} and $f_{a\overline{b}}$ partition the space defined by f_a. Hence, by appendix A, the model including f_a and f_{ab} is equivalent to the model including f_a and $f_{a\overline{b}}$. Hence the second check above is computing the count of $f_{a\overline{b}}$ (and $f_{\overline{a}b}$), because we don't want the count of that "feature" to be below our minimum count threshold of k. Note that this requirement also imposes an implicit condition that $\#f_a > 2k$ and $\#f_b > 2k$ if f_a and f_b are going to be part of a compound feature. Finally, note that in our tests on compound features, we used $k = 3$, the same threshold which we set for selecting atomic features.

After filtering compound features by count in this way, we were still left with a pool of roughly 139,000 compound features created from our initial pool of 22,000 atomic features. This is far too many features for the maximum entropy parameter estimator which we were using [43] to handle. The task at this point is to select the features from this vast pool which would most enhance our 22,000-feature atomic model. There are several ways of doing this [4] [3], but we are using a technique first proposed in [42] in which we choose features by comparing the model's expectation of how often they should occur in the training corpus with their empirical probability of occurrence. In particular, for a given candidate feature g_j, and a baseline maximum entropy model P_M, we are looking at the quantities:

$$K_j = \sum_{h,f} \tilde{P}(h) P_M(f|h) \cdot g_j(h, f) \tag{6.1}$$

$$\tilde{K_j} = \sum_{h,f} \tilde{P}(h, f) \cdot g_j(h, f) \tag{6.2}$$

By equation 3.13, if g_j were incorporated into P_M, then K_j and $\tilde{K_j}$ would be equal. Thus a large difference between the two values would indicate that the

model would be significantly altered by incorporating the feature. In our work we tried looking at the both the absolute value of the difference between the two quantities (the "expectation discrepancy") and the ratio between the larger and the smaller of the two quantities (the "expectation ratio").

The advantage of this approach is that these ratios or differences can be calculated in a single pass through the training corpus whereas the method of [4] requires 4 - 5 passes through the corpus [3]. It is also trivial to implement, unlike the method used in [3] and [38].

Finally, we found that we had to manually delete certain classes of features by hand which fired too frequently and co-occurred in clusters. Forty or more features firing on the same history/future combination causes numerical difficulties when attempting to solve equation 3.33, which is the key equation in the improved iterative scaling method of computing the α weightings [20] which is used in the M.E. toolkit which we were using [43]. In particular, we had to delete all compound features involving the "uncapitalized word" feature which was illustrated in table 5.1. Since we had atomic features which fired when w_{-1}, w_0, and w_1 were uncapitalized, and these three features activated with very high frequency, they tended to appear very often in compound features. When we had a string of three or more uncapitalized words, compound features involving any one of the three could fire simultaneously. This often led to us exceeding the 40-feature limit.

It is clearly unfortunate for a feature-selection technique to require any manual intervention at all, but we believe that this step could be automated if necessary. One could, for instance, scan the "events" file described in section 4.2.2 for history/future combinations on which more than j features were active ($j = 40$ is probably appropriate). Of the features, F which occurred in these $\langle h, f \rangle$ combinations, we could choose to remove features from the feature pool by the following two-step method:

- Determine the atomic features $a(h, f)$ which occurred most often as the component of the compound features in F.
- Remove all compound features from the feature pool which contains a
- Repeat until no $\langle h, f \rangle$ pair has more than j active features

This procedure would be essentially the same as our manual deletion process.

We tried a number of experiments with models which incorporated compound features with varying levels of absolute expectation discrepancies. We also built models in which compound features were selected for inclusion in a model based on whether their expectation ratios exceeded a certain threshold, where we tried thresholds of 1.05, 1.1, and 1.2. Unfortunately, none of these experiments yielded

models which showed consistent, significant improvements over a baseline atomic-feature-only model, so our models are only using "atomic" features pending further research.

Chapter 7

Results–English

MENE's maximum entropy training algorithm gives it reasonable performance with moderate-sized training corpora or few information sources, while allowing it to really shine when more training data and information sources are added. This chapter will look at some of these results and compare them to some results from other systems. We will also look at how MENE's performance was affected by the presence or absence of various features from the feature pool.

7.1 Other Systems' Results at MUC-7

NYU entered the MENE system in the MUC-7 evaluation (discussed in section 1.4), where it garnered a respectable ranking of 4th out of the 12 systems entered in the evaluation. All statistics given in this chapter are "F-measures", which we defined in figure 1.1. MUC-7 results from the top five systems are summarized in table 7.1.

Institution	Technique	F-measure	Precision	Recall
Edinburgh/LTG	M.E./hand-coded hybrid	93.39	95	92
IsoQuest, Inc.	Hand-coded	91.60	93	90
BBN	Hidden Markov Model	90.44	92	89
NYU	M.E./hand-coded hybrid	88.80	93	85
U. of Manitoba	Hand-coded	86.37	87	85

Table 7.1: Top five systems at MUC-7

As we can see, three of the top five systems made use of statistical modeling in some way. Note also that the median F-measure at the evaluation was 85.57 and that the lowest score was 69.67. The MENE system used in the evaluation was

a system which took in output from the Proteus handcoded system as one of its inputs as described in section 5.5.

7.2 MENE's MUC-7 and System Combination Results

Table 7.2 shows MENE's performance on the within-domain corpus from MUC-7's dry run as well as the out-of-domain data from MUC-7's formal run. All of these MENE systems were trained on 350 aviation disaster articles (this training corpus consisted of about 270,000 words, which our system turned into 321,000 tokens).

Systems	Dry Run F-Meas.	Dry Run Prec.	Dry run Recall	Formal F-Meas.	Formal Prec.	Formal Recall
MENE (ME)	92.20	96	89	84.22	91	78
Proteus (Pr)	92.24	95	90	86.21	88	84
Manitoba (Ma)	93.32	94	92	86.37	87	85
IsoQuest (IQ)	96.27	98	94	91.60	93	90
ME + Pr	95.61	97	94	88.80	93	85
ME + Ma	95.49	97	94	88.91	92	86
ME + IQ	96.55	98	95	91.53	94	89
ME + Pr + Ma	96.48	97	95	90.34	93	88
ME + Pr + IQ	96.78	98	96	92.05	95	89
ME + Ma + IQ	96.81	98	95	91.84	95	89
ME + Pr + Ma + IQ	97.12	98	96	92.00	95	89

Table 7.2: System combinations on unseen data from the MUC-7 dry-run and formal test sets

Note the smooth progression of the dry run scores as more information is added to the system. Also note that, when combined under MENE, the three weakest systems, MENE, Proteus, and Manitoba outperform the strongest single system of the group, IsoQuest's. Finally, the top dry-run score of 97.12 from combining all three systems seems to be competitive with human performance. On a different set of data, the MUC-7 formal run data, the accuracy of the two human taggers who were preparing the answer key was tested and it was discovered that one of them had an F-Measure of 96.95 and the other of 97.60 [32]. Although we don't have human performance measures on the dry run test set, it seems that we have attained a result which is at least competitive with that of a human.

The formal evaluation involved a shift in topic which was not communicated to the participants beforehand–the training data focused on airline disasters while the test data was on missile and rocket launches. MENE faired much more poorly on this data than it did on the dry run data. While our performance was still reasonably good, we feel that it is necessary to view this number as a cross-domain portability result rather than an indicator of how the system can do on unseen data within its training domain. In addition, the progression of scores of the combined systems was less smooth. Although MENE improved the Manitoba and Proteus scores dramatically, it left the IsoQuest score essentially unchanged. This may have been due to the tremendous gap between the MENE- and IsoQuest-only scores. Also, there was no improvement between the MENE + Proteus + IsoQuest score and the score for all four systems. We suspect that this was due to the relatively low precision of the Manitoba system on formal-run data.

We also did a series of runs to examine how the systems performed on the dry run corpus with different amounts of training data. These experiments are summarized in table 7.3.

Systems	425	350	250	150	100	80	40	20	10	5
MENE	92.9	92.2	91.3	90.6	89.2	87.9	84.1	81.0	76.4	63.1
MENE + Proteus	95.7	95.6	95.6	94.5	94.3	93.4	91.7			
MENE + Manitoba	95.6	95.5	95.3	94.9	94.5	94.2	93.1			
MENE + IsoQuest	96.7	96.6	96.7	96.6	96.1					
ME, Pr, Ma, IQ	97.4	97.1								

Table 7.3: Systems' F-measures on dry-run data with different numbers of articles

Note the 97.4 all-systems result which we achieved by adding 75 articles from the formal-run test corpus to the basic 350-article training data. In addition to being an outstanding performance figure, this number shows MENE's responsiveness to good training material. A few other conclusions can be drawn from this data. First of all, MENE needs at least 20 articles of tagged training data to get acceptable performance on its own. Secondly, there is a minimum amount of training data which is needed for MENE to improve an external system. For Proteus and the Manitoba system, this number seems to be about 80 articles, because they show a degradation of performance at 40. Since the IsoQuest system was stronger to start with, MENE required 150 articles to show an improvement. Note the

anomaly in comparing the 250 and 350 article columns. Proteus shows only a very small gain and IsoQuest shows a deterioration. These last 100 articles added to the system were tagged by us at NYU, and we would humbly guess that we tagged them less carefully than the rest of the data which was tagged by BBN and Science Applications International Corporation (SAIC).

7.3 Upper Case Results

At the time of the MUC-7 evaluation, we also ran MENE against all-uppercase data as a first step towards testing its performance on spoken English. We trained the system on an uppercase version of our basic 350-article training corpus. We used all the same features in our upper case model as we did in our MUC-7 model, except that we deleted all of our capitalization-sensitive features (the "initial-cap", "all-cap", "all-lowercase", and "internal-cap" features from section 5.1). Note that our lexical features are always case-insensitive, so they did not have to be modified at all.

Upper case results are shown in table 7.4. BBN and IsoQuest results are taken from their MUC-7 papers [37] [29]. Note that the MENE-Proteus system scored higher on this data than did the IsoQuest system even though IsoQuest scored significantly higher than the NYU system on mixed case data. BBN, an automatically trained system, maintained and widened its formal-run lead over NYU on this data.

System	Mixed-case Formal	Mixed-Case Dry Run	Upper-Case Formal	Upper-case Dry Run
MENE	84.22	92.20	77.98	88.19
MENE + Proteus	88.80	95.61	82.76	91.38
BBN	90.44	95.1	87.6	?
IsoQuest	91.60	96.27	81.96	?

Table 7.4: Comparative upper case results

Our analysis of this data is firstly that this is an illustrations of the portability advantage that automatically trained systems have over hand-coded systems, even very good ones like IsoQuest's. Secondly, although this MENE system was trained on uppercase data, we did not work very hard at optimizing the feature pool for upper-case, so we suspect that there is room for improvement in this result. Finally, note that our system showed significantly more deterioration on all-uppercase data than did BBN's. We are not sure what the cause of this is.

7.4 Contributions of the Different Feature Classes

We have analyzed the contribution made to the system by the different classes of features included in the model. These results are summarized in table 7.5.

Ref Num	System	Formal Run Result	Dry Run Result	Note
1	Basic MUC-7 MENE-only System	84.22	92.20	
2	MENE with only lexical and section features	77.60	88.13	
3	MENE without dictionary features (the above system + binary features)	83.38	91.71	
4	MENE without lexical features	58.35	63.90	
5	MENE without binary features	79.61	88.60	
6	MENE without section features	84.22	92.33	
7	MENE-only w/ref resolution (Trained on "cheating" data)	86.26	93.20	A
8	MENE-only w/ref resolution (Trained on "rotated" data)	86.56	93.56	B
9	MENE + Proteus	88.80	95.61	
10	MENE + Proteus w/ref resolution (Trained on "rotated" data)	90.25	95.75	B

A These reference-resolution features were trained on the output of system 1 when run against its own training data. See 5.6 for details.

B Reference-resolution features were looking at MENE output trained on the output of 5 models trained on 80% of the training corpus and tested on 20%, rotated appropriately.

Table 7.5: Comparative feature contributions

We found this data to be very interesting and drew several conclusions from it. First of all, section features turned out to be not useful, as can be seen by comparing systems 1 and 6 in the chart. We speculate on the reasons for this in section 5.3.

Secondly, a comparison of systems 1 and 3 shows that the contribution of our dictionary features was small, although not non-existent. We think that this stems from the fact that these features have essentially no impact on our predictive ability for any word which has an associated lexical feature, as we showed in section 5.4. Since every word which occurs three or more times is included in our vocabulary

and almost every vocabulary word will have a lexical feature, we can expect that dictionary features will be useful for making predictions on only a relatively small percentage of the tokens of a typical test-corpus.

Thirdly, the omission of feature classes hurt performance on the out-of-domain formal-run corpus more than the in-domain dry-run corpus, with the exception of the no-lexical-feature model (system 4). We think that this can be easily explained by looking at the problem from the other perspective. For instance, in system 3, the error-rate in the formal-run was roughly twice as much as in the dry run. Hence it is reasonable that the improvement from adding dictionary features to the system was 71% higher on the formal run since there were twice as many errors available for correction. While these numbers don't completely correlate, we think that they are close enough to suggest that nothing unusual is going on here. As for system 4, the lexical features would be most vulnerable to a shift in domain, so it seems reasonable that, in this case, the fall-off from omitting lexical features was greater on the within-domain dry-run corpus than it was on the out-of-domain formal-run corpus.

Fourthly, the addition of reference-resolution features helped the system tremendously. In fact, adding these features reduced the difference between the MENE-only and the MENE + Proteus (without MENE reference resolution) systems by 40 - 51%. This conformed to our intuition that a lot of what Proteus was "bringing to the table" in the combined MUC-7 system was a reference-resolution engine which was lacking in the basic MENE system. It should also be noted that the MENE + Proteus system benefited from the addition of the reference-resolution features, particularly on the formal run where the addition of these features gave us the best results obtained by an NYU-only system. This would seem to be an indication that the reference-resolution method we are using (which we discussed in section 5.6) is superior to the one being used in Proteus, although it is also possible that the MENE-Proteus-reference-resolution system is benefiting from the ability to propagate names which MENE identified in the first pass but Proteus did not.

Finally, training the reference resolution features on rotated data yields a small, but significant increase in performance. This is in line with informal experiments we had when training the MENE + Proteus system. During the MUC-7 development process both MENE and Proteus were being developed in parallel. As Proteus was incrementally improved we occasionally would run a MENE + Proteus system trained on the training-corpus-output of a new version of Proteus against the test-corpus-output of an older version of the system. This consistently produced worse results than a comparable system trained and tested against the older version. The reverse, however, was not true. Training on old Proteus output and testing on new Proteus output performed better than just using the old output. We think that these results are understandable when one considers that MENE needs to "learn" the sorts of errors that a given feature will make so that

it can "compensate" by strengthening other features. A system which is more accurate than expected will not cause any harm, but if a system makes mistakes during testing which it didn't make during training, the other features probably will not be able to compensate.

When the MUC-7 MENE-only system was run against its own training data (this is "cheating"), it scored an F-measure of 96.14. On the other hand, the score for the concatenation of the 5 20%-test-corpora generated by splitting the training corpus 80/20 with 80% for training and 20% for testing was F=90.54. This number is intermediate between the 84.22 basic system score on formal-run data and the 92.20 score on dry-run data, so the rotated output is clearly representative of the system's performance. Given the reasoning of the previous paragraph, it makes sense that the system trained on this rotated data performed better than the system trained on cheating data.

7.5 Comparison with BBN's HMM System

Conditions	BBN: Identi-finder	MENE	MENE with Ref Res	MENE + Proteus	MENE + Proteus + Ref Res
Train: Official training data Test: Dry run (In domain)	92.5	89.17	91.63	94.30	94.02
Each site trained on it's own data. Test: Dry run	95.1	92.20	93.56	95.61	95.75
Train: Same as above Test: MUC-7 formal run	90.44	84.22	86.56	88.80	90.25

Table 7.6: Comparison of BBN and NYU statistical systems

Table 7.6 gives a comparison of BBN's HMM-based Identifinder system [37] with NYU's MENE-Proteus, basic MENE, MENE-with-reference-resolution, and MENE-Proteus-Ref-Resolution systems on different training and test sets. The first thing to note in this data is that the MENE + Proteus system had a higher score on the dry run data than Identifinder but scored lower on the formal-run. We are not sure why MENE-Proteus was hurt more badly by the evaluation-time switch from aviation disaster articles to missile/rocket launch articles, but suspect that it may have been due to Identifinder's greater quantity and quality of training data. BBN used 790,000 words of training data to our 321,000. The quality advantage may have come from BBN's technique of selecting sentences from a larger corpus for their annotators to tag which were chosen so as to increase the

variety of training data. An advantage in the quantity and breadth of training data would be most likely to be felt on out-of-domain data since a system training on this data would tend to be more robust.

The other interesting thing about this chart is that adding reference resolution features to MENE has nearly closed the gap between MENE and Identifinder when trained on the official training data. Identifinder dynamically updates its vocabulary during decoding when person or organization names are recognized, giving the system a form of long-distance reference resolution. It would seem that reference-resolution accounted for much of Identifinder's performance advantage over the basic MENE system. Note also that the addition of MENE's reference resolution features has brought the MENE-Proteus system up to rough equality with Identifinder on formal-run data when both systems are trained on all of their own data (an official-run type of situation). Hence, Proteus' presence compensated for the fact that BBN had roughly 2.5 times as much training data as NYU.

Another characteristic of Identifinder is that it implicitly predicts named entities based on consecutive pairs of words rather than based on single words, as is done in MENE, because each type of name has its own bigram language model. In the decoding process, the Viterbi algorithm chooses the sequence of names which yields the highest joint probability of names, words, and features associated with each word. As we discussed in section 5.8, MENE's features work individually to make a prediction and we have thus far been unsuccessful in adding an ability to predict based on patterns using our "compound features". It is possible that some of Identifinder's residual advantage of 0.9F in an apples-to-apples comparison with the MENE + Reference Resolution system comes from making its predictions based on pairs of words.

Chapter 8

Experiments with Japanese

As a test of MENE's portability, the author ran a series of experiments with Japanese-language text. These experiments were run against data from the foreign language evaluation known as MET-2 which was held at the same time as the MUC-7 evaluation. The MET-2 evaluation followed the same format as the English-language MUC-7 conference described in section 1.4 and had the same set of domains (aviation and other disasters for the training data and dry run and missile launches for the formal evaluation). Our tests were run subsequent to the MUC-7/MET-2 conference, so they were not reported there. In addition, we participated in the Information Retrieval and Extraction Exercise (IREX), a Japanese-language named entity evaluation. We report these results in this chapter and will deliver a more complete report at the upcoming conference.

8.1 Data Pre-processing and System Architecture

Japanese-language text posed a special problem for MENE since MENE expects to receive words as a stream of individual tokens. Since Japanese doesn't use any word spaces, we relied on the "Juman" morphological analyzer [34] to pre-process the text. In addition to tokenization, the Juman system also gave us part-of-speech information on each token.

The work in this chapter would not have been possible without the assistance of our colleague, Dr. Satoshi Sekine. Sekine is a native speaker of Japanese whereas this author can not read any Japanese whatsoever. Hence, since we are arguing that MENE has very little dependence on the implementor's linguistic intuition, we think that it is important to be specific that Sekine's contribution to this work was limited to building the interface to the Juman system, tagging the types of characters in each token, building the Japanese-language dictionaries, encoding

Japanese text as ASCII alphabetic characters for the author's convenience, creating SGML-marked text from MENE output, and building an interface to the scoring routines which reported F-measures on test corpora. While this is a substantial amount of work, except for the alphabetic encoding, all of this was pre-existing code from Sekine's decision-tree-based named entity system [48] which was discussed in section 2.3.

In addition to the above, Sekine installed an *ad hoc* patch of Juman's tokenization to fix some problems with Juman's treatment of certain idiomatic Japanese expressions in the IREX data (for instance, Juman treated the phrase "visit-Japan" as one word, whereas for IREX purposes, we would want to tag "Japan" as a location separate from "visit"). This patch tagged some entities outside of the MENE process and boosted our IREX scores by about 1.3 F-measures.

The basic architecture of the system was identical with that of the English-language system except for this pre-processing stage and for the use of Sekine's SGML-generating module at the very end of the decoding process. In particular, the only steps of the training algorithm in figure 4.1 which differed between our English and Japanese systems were steps 2 (Training corpus tokenization) and 4 (History view precomputation).

Given the framework established by the decision-tree system and the great deal of flexibility and modularity which we had established in our English-language MENE system, we were able to port the system to Japanese in just three person-days. Most of the time spent doing the port was devoted to building an interface to the files which we received from the decision-tree system's pre-processing programs. All of the major training-time and run-time programs shared the same code between the English-language and Japanese-language systems. The core modules lacked even a "japan" run-time flag since all the factors which differed between English and Japanese were captured in the features and history-views.

8.2 Features Used in Japanese MENE

The features used in the Japanese system were similar to those used in the English-language system. We will discuss each of them in turn. Results showing how system performance was affected by the inclusion of different set of features are given in tables 8.1 and 8.2.

- Lexical Features: Lexical features were exactly the same as the English-language features reported in section 5.2 except that they were looking at a window of $w_{-1} \ldots w_1$ rather than the $w_{-2} \ldots w_2$ window used by the English MENE system. This smaller window was selected based on experiments which failed to show a consistent improvement in performance when the w_{-2} and w_2 words were examined.

The "words" that the lexical features were looking at were tokenized by Juman, as described earlier, and thus were subject to tokenization errors. Of course there was the possibility of tokenization errors in the English-language system too (see section 4.2.1), but the problem was much more serious in Japanese. In any language, tokens which contain some characters inside and some outside a named entity will inevitably lead to errors.

The tokens on which MENE operated were Roman-alphabet encodings of Japanese-language text which guaranteed a one-to-one mapping between an encoding and a string of Japanese characters. This encoding was unintelligible to a speaker of either English or Japanese.

- Character type: Features which fire based on whether a given word (in a $w_{-1} \ldots w_1$ window) is a particular type of character (i.e. Kanji, Katakana, etc.) or a mixture of different characters (i.e. both Kanji and Katakana). There were 30 different possible characters and combinations. The character type of each token was determined by Sekine's pre-processing program. Note that character type is a useful predictor of named entities in Japanese. The Katakana characters, for instance, form an alphabet which is often used for transcribing foreign names.

- Part of speech: Features firing on a token's part of speech. Part of speech was determined by Juman, which assigns every word a "major" and "minor" POS category. Some "minor" POS categories were discarded by Sekine's pre-processing program as unimportant, so MENE saw every word identified as belonging to one of 31 major or major/minor categories. Clearly Juman could misidentify a word's POS, so this data was somewhat noisy.

- Dictionary: We relied on the dictionaries which Sekine created for his decision-tree system [48]. The form that the dictionaries took was consequently somewhat different from the form of our English-language dictionaries. The dictionaries consisted of word lists which would predict any one of the seven NE categories based on a word being present in the w_{-1}, w_0, or w_1 position. Hence, there were 21 dictionaries ($7 * 3$). For instance, the Japanese system had dictionaries of "organization prefix", "organization suffix", and "organization-current-word". By contrast, all of the English MENE dictionaries were primarily designed to make predictions about w_0, but we had multiple dictionaries whose goal was to predict the same entity (for instance, we intended both the "corporate names" and the "colleges and universities" dictionaries to make the prediction that w_0 is a an organization). Of course, as with any feature, MENE would generate a feature predicting, for instance, "person_unique" for w_0 given a Japanese dictionary-tag

of "organization-current-word" for w_0 if this situation happened three times or more in the training corpus. One would assume, of course, that such a feature would be assigned a lower α value in training than one predicting "organization_unique" given a dictionary tag of "organization-current-word" to w_0.

8.3 Japanese Results–MET-2 data

Our results on MET-2 evaluation data are summarized in table 8.1. While this system was not completed in time to be entered in the MET-2 evaluation, had the system been entered, its F-measure of 83.80 would have allowed it to finish in a virtual tie for second-place with a system from NTT Data [21] in a four-system field. The top-ranked system from Oki Electronics [22] achieved a much-higher F-measure of 90.54. While it is not entirely clear why the Oki system was so strong, it appears that a significant portion of the advantage was due to Oki's superior usage of dictionaries [23]. In particular, Oki was the only site to have given their system a list of heavenly bodies (i.e. "the moon", "Mars", etc.), which appeared with great frequency as locations in the missile/rocket-launches which made up the MET-2 test corpus.

Heavenly bodies are an example of a situation where a hand-coded system could have a significant advantage over a statistically-trained system like MENE. The constructor of a hand-coded system can refer to the document giving the named entity task definition [14], see that a particular category like "heavenly bodies" is listed there as a type of location, and add an appropriate word-list into the system. The constructor of a statistically-trained system such as MENE could similarly include such a word-list, but it could be that in a domain like the vehicular disaster domain which made up the MENE training corpus, there wouldn't be enough references to adequately train a specialized word-list like one for heavenly bodies. So it is clear that a handcoded system can have an advantage over a statistical system when the training data does not match the testing data.

Table 8.1 shows the performance of the basic MENE system which achieved the highest results. The table also shows results from systems which exclude different classes of features to give an indication of the degree to which the system was relying on different types of information.

8.4 Japanese Results–IREX data

We also tested MENE on Japanese corpora from "IREX". IREX is a Japanese-language-only named entity evaluation (or "exercise") similar to the MUC-7/MET-2 evaluation. IREX had a "dry run" test in November, 1998, which was followed

System	Formal Run F-Measure	Precision	Recall
Basic MENE-MET2 System	83.80	91	78
No lexical features	79.97	87	74
No dictionary features	81.56	89	75
No character type features	83.19	90	77
No part of speech features	83.18	89	78

Table 8.1: MET2 results with different sets of features

by a formal evaluation on May 13, 1999. More detailed results on MENE's performance will be reported at the IREX conference September 1 - September 3, 1999.

The primary differences between the MET-2 and the IREX evaluations were the following:

- The domain of MET-2, as stated above, was focused on the specific topics of vehicular disasters and missile launches. IREX, however, included general Japanese newspaper text as its "domain". This general domain included general news stories, business, entertainment, sports, and other topics. There was also a domain-specific portion of IREX for which we submitted results from our IREX model trained on the general domain. We did not train a domain-specific system due to time constraints.

- MET-2 includes an eighth named entity class, that of "artifact", which was defined as follows [47]:

  > Artifacts are the names of proper things, including abstract and concrete objects. They include merchandise, books, publications, laws, treaties, theories, buildings, prizes, stocks, services, TV shows, radio programs, movies, etc.

  We think that it is indicative of MENE's flexibility that we were able to rapidly add an eighth class of named entities to the system without any alterations to MENE's core code. The additions of four futures (artifact_unique, artifact_start, artifact_continue, and artifact_end) to the future vocabulary file and a corresponding revision to the file which defines permissible future-to-future transitions (i.e. artifact_continue can not follow person_start) were sufficient to expand the total number of futures from 29 to 33 in the system's core C++ modules.

- The MET-2 training corpus consisted of only 73,000 tokens (our "evaluation" system was trained on an 80,000-token corpus consisting of the training data plus the dry-run data). The IREX training corpus was much larger, however, consisting of 294,000 tokens. The MET-2 dry-run training corpus was thus almost identical in size to the 80-article (80,000-token) English-language training corpus reported in table 7.3. The IREX training corpus, however, was close to the size of our baseline 350-article (321,000-token) MUC-7 training corpus. Since table 7.3 showed a tight link between training corpus size and F-measure, we have to bear in mind this discrepancy in training data when comparing MET-2 and IREX results.

MENE's results on the IREX evaluation's dry-run and formal-run corpora are summarized in table 8.2 with statistics on how the system performed when different feature classes were excluded, similar to what we showed in our MET-2 results in table 8.1. All statistics in this table are for a system trained on the full 294,000-token IREX training corpus.

System	Dry Run F-Measure	Formal F-measure	Formal Precision	Formal Recall
Basic MENE-IREX System	74.68	77.37	83.45	72.12
No lexical features	65.32	69.05	76.70	62.78
No dictionary features	73.09	75.10	82.37	69.01
No character type features	74.32	76.68	83.36	70.99
No part of speech features	72.56	77.22	82.74	72.38

Table 8.2: IREX results with different sets of features

Table 8.3 gives comparative results for MENE at the IREX evaluation. MENE's F-measure of 77.42 was the third highest at the evaluation from a field of fourteen participants. This result was significantly behind the scores of the top two systems, but significantly ahead of the fourth ranked system and well above the median. Two other facts are interesting to note here. First of all, MENE's relative performance actually improved on the domain-specific "arrest" corpus even though we did not train a system geared specifically to that domain. Secondly, note that MENE's performance improved from our tests on dry-run data to those on the formal-run data. The latter effect was very much different from our experience in MUC-7, where we saw a significant decline between our preliminary experiments on dry run data and the formal evaluation (as was shown in table 7.2). We think that this effect was probably due to greater consistency between the dry-run and formal test corpora than was the case in MUC-7. It is also possible that the IREX

formal test corpora was somewhat easier than the dry run corpora due to the presence of a number of relatively easy obituary articles [47].

By the rules of the IREX evaluation, the administrators of the evaluation only identify the participating systems by ID, so we don't currently have any information about what techniques were used by the other systems. However, we believe that our third-place rank in a Japanese-language exercise in which every other system was constructed by a team which had a native speaker of Japanese as its lead designer [47] is a very favorable result. It certainly is an indicator of MENE's cross-lingual portability and lack of dependence on the constructor's linguistic intuition.

Institution	Technique	F-measure on general corpus (with rank)	F-measure on arrest corpus (with rank)
Unknown		83.86 (1)	87.43 (1)
Unknown		80.05 (2)	78.05 (5)
NYU: MENE	Maximum Entropy	77.37 (3)	85.02 (2)
Unknown		75.30 (4)	77.61 (6)
Unknown		74.82 (5)	81.94 (3)
Median Score	n.a.	71.15 (7.5)	74.73 (7.5)

Table 8.3: IREX comparative results

8.5 Experiments With Long-Distance Reference Resolution

Extensive experiments with both the MET-2 data and the IREX data failed to turn up any consistent improvements from the use of long distance reference resolution features. These features were using the same methodology as the ones which we used very successfully in the English MENE system, as described in section 5.6 and as originally described in [35]. In particular, we used the same system for identifying acronyms that we used in English–we looked for words which were composed of the first characters of a sequence of words which were identified elsewhere as an entity, allowing for words to be omitted from the sequence. While this author doesn't have the linguistic knowledge to judge the merit of this approach, it was the opinion of two native speakers of Japanese [45] [33] that both the acronym and full-word reference resolution techniques were plausible methods of detecting Japanese references.

We tried a number of different approaches to incorporating these features in the system, all of which failed to display a consistent improvement over a baseline Japanese system which did not include reference resolution, as shown in figure 8.4. Among the tests we ran were experiments in which we trained the system on its own training data (which we label as training on "cheating" data in figure 8.4) and training on a rotated training corpus (labeled as "rotated" in figure 8.4).

Training Corpus	Test Corpus	Language	Pass-1 F-Score	Pass-1 Precision	Pass-1 Recall	Pass-2 F-Score
MUC-7 Rotated	dry run	English	92.20	96	89	93.56
MUC-7 Rotated	formal	English	84.22	91	78	86.56
MET-2 Cheating	formal	Japanese	83.80	91	78	82.48
IREX Cheating	dry run	Japanese	74.46	79.26	70.22	73.43
IREX Rotated	dry run	Japanese	74.46	79.26	70.22	74.25

Table 8.4: Japanese reference resolution experiments

Due to these negative results we decided not to include reference-resolution features in our official entry in the IREX exercise. We remain puzzled as to why an approach which was so successful in English should have had such poor results in Japanese. We think that answering this question will require the assistance of a native speaker of Japanese, but we can offer a few general hypotheses:

- As mentioned in section 5.6, our English reference-resolution features made use of a dictionary of single-token function words which allowed us to create reference-resolution features which only fired on single tokens which were not function words (and thus avoid firing on words like "the" or "of" which appeared as parts of named entities in the first pass). Our Japanese-language system may have been hampered by our lack of an equivalent list of Japanese function-words.

  In an attempt to generate such a list in the absence of any linguistic knowledge, we tried using lists of the most frequently appearing words in our training corpus, assuming that any names appearing on this list would be so common (i.e. "Japan") that we would not need the aid of reference-resolution features to properly identify them. Unfortunately, this crude system proved unsuccessful. It remains to be seen whether a linguistically accurate list of function words would improve the system.

- Reference-resolution may be less important in Japanese than in English. In English, for instance, standard newspaper text would give a first reference

to the author's name as "Andrew Borthwick" while subsequent references might be simply to "Borthwick", a very difficult case to pick up without reference resolution. In Japanese newspaper text, however, subsequent references would be to "Borthwick-san", a case which MENE would identify very easily with the aid of a feature firing on "san".

- The simple approach which worked in English may be the wrong approach in Japanese.

8.6 Comparison of Maximum Entropy and Decision Tree Approaches

The existence of the MENE system and Sekine's decision tree system as two approaches to Japanese NE recognition gives us a rare opportunity to make a precise side-by-side comparison of the utility of two different statistical methods applied to the same problem. This comparison is facilitated by the fact that, with the exception of lexical features, all the information that MENE used was also available to the decision tree system. Unlike most attempts at comparing two approaches (such as with the comparison of the maximum entropy and Hidden Markov Model approaches described in section 7.5), this comparison is not complicated by questions of the quantity and quality of training data, the cleverness of the processing of the data, or of the availability of other resources to the two systems such as dictionaries. We are able to make some direct comparisons between the two statistical methods as represented by our reasonably straightforward approaches using two off-the-shelf toolkits [39] [43].

Determining which approach would be more successful is primarily an empirical question. Decision trees work well in cases in which "features" (or "questions" in the DT world) perform differently as conjuncts than they do individually. For instance, in named entity recognition on English text, a feature saying that w_0 is capitalized is strongly predictive that w_0 is a named entity. Similarly, w_0 being in a headline is mildly predictive that w_0 is an NE. However, capitalization is conditionally independent of w_0 being an NE given that w_0 is in a headline, so a maximum entropy "capitalization and headline" feature would strongly predict "other" given the presence of the other two "atomic" features in the system.

Finding these compound features is not an easy task for an M.E. model, as shown by the results of sections 5.8 and 6.2. Their use, however, is inherent in decision trees, so if feature conjuncts frequently behaved differently from the way that features behave in general, a decision-tree model is likely to perform better than an ME model.

Maximum entropy excels, however, in situations where it is reasonable to weight

the combination of two features by taking the product of their weights (as shown in equation 3.3). For instance, the fact that w_0 is an unknown word (not in the vocabulary derived from the training corpus) is mildly predictive that w_0 is a location because obscure locations (like other named entities) frequently turn up in newspaper text. Similarly w_{-1} = "to" is mildly predictive that the next word is a location, although this is far from a sure thing. The two-word phrase "to unknown_word" does not seem like a special case to this author's intuition, so it seems reasonable that the two features should exhibit their default behavior when they activate together.

We have run several experiments which attempt to directly compare the two approaches, which are summarized in table 8.5. In this chart, the column "MENE with only DT Features" gives results from a MENE system which did not use any lexical features and consequently was operating with the same information sources which were available to the decision-tree system. The next column "MENE with all features" is for MENE systems which include the lexical features. The column "best score" gives the best known score on the task with that set of training data.

Taking the three rows of the chart in turn, the first line of the chart shows results on the MET-2 formal-run data which indicate that given a moderate-sized training corpus, maximum entropy and decision trees have roughly equivalent results when they both have the same information to work with. Maximum entropy has a significant advantage, however, when lexical features are added to the system, with the M.E. system outscoring the DT system by over four F-measures. Although lexical features made up over 80% of the total number of features in MENE's MET-2 system, adding them to the system was an almost trivial task for us (due in large part to some technical assistance from Sekine). On the other hand, it is not clear how lexical features can be added to a decision-tree system. A naive implementation in which lexical "questions" are added which are of the form "Is the [current|previous|next] word x" (which is essentially the way they were implemented in MENE) produced a system which exceeded the capacity of the decision-tree toolkit [45]. Furthermore, even if it was technically feasible to add these features to a decision-tree, it is quite likely that they would cause data fragmentation problems, as discussed in section 2.3. It remains to be seen whether a more sophisticated approach such as the clustering techniques used in [31] would yield better results.

The second row of the chart is interesting in that we had only a very small training corpus to work with. With this training corpus, MENE outperformed the DT system even when it didn't use any lexical features. This result from training on sparse data corroborates our results with the lexical data in that a system which contains individual words as features is going to have inherent difficulties with data sparseness since even the more common words appear relatively infrequently in a training corpus. With only 14,000 words in the training corpus, the dictionary and

part-of-speech features will also encounter sparse data problems, so it seems that we can conclude that maximum entropy is the better approach when dealing with problems of sparse data.

The third line of the chart illustrates the opposite problem: in this case we have a great deal of training data. The full MENE benefits tremendously as its performance increases by over 12 F-measures. The system without lexical features shows a much smaller gain. Unfortunately, the C4.5 decision-tree toolkit [39] crashes on this volume of data. It remains to be seen how the two systems would compare under these circumstances.

Training Data	Testing Data	Num. of Training Words	MENE w/only DT Features	MENE w/all features	DT Score	Best Score
Official MET-2 + dry run test	MET-2 Formal	80,000	79.97	83.80	79.49	90.54
IREX dry run training	IREX dry run	14,000	61.67	62.39	58.93	68.23
Full IREX training	IREX dry run	294,000	65.32	74.68	Crashes	?

Table 8.5: Comparison of MENE with DT system

Chapter 9

Conclusions

We will conclude this work by assessing the practicality of real-world applications of the MENE system, speculating on its ease of portability to other language and outlining some future avenues for research on maximum entropy approaches to named entity recognition.

9.1 Assessment of the Utility of a Maximum Entropy Approach

As shown by MENE's results on the IREX Japanese named-entity evaluation, MENE as a stand-alone system is able to achieve levels of accuracy in named-entity tagging which are solidly competitive (well above the median), but are lower than the scores of the very best handcoded systems. If it were entered as a stand-alone system (without the Proteus input), MENE would have scored roughly at the median at the MUC-7 evaluation, although with the later addition of the reference resolution features discussed in section 5.6, we would have ended up about 2 F-measures above the median. Compared to the other major current statistically-trained system, BBN's Hidden-Markov-Model-based "Identifinder" system, we seem to trail slightly when differences in the quantity of training data are accounted for.

Consequently, as a stand-alone system, MENE's main current contribution to the field of information extraction is that it is highly portable. The results on Japanese show that it can be used to achieve good results in a language which the implementor does not speak. Even in English, the aspect of the system which required the most linguistic intuition from the implementor (the dictionaries), accounted for less than 0.9F of our score on MUC-7 data, as shown in table 7.5.

While these stand-alone results are promising, we feel that the results on combining MENE with other systems are some of the most interesting because they

seem to be the most widely applicable. To summarize the results, we found that the Proteus, University of Manitoba, and IsoQuest systems were all improved by being combined with MENE on the within-domain MUC-7 dry-run data and all but the Isoquest system improved on the out-of-domain formal-run data. Furthermore, when we combined all three systems, we achieved near-human-level results on within-domain data and combined the 2nd-, 4th-, and 5th-ranked systems to significantly improve the 2nd-ranked system's score.

Since MENE requires no specific customization to accept a new named entity system as an input, we would speculate that most handcoded systems could achieve higher input by using MENE as a sort of post-processor whereby their output is an input to MENE, which produces the final, refined output. Furthermore, it seems likely that the way to achieve the highest possible accuracy is to combine as many systems as possible under MENE's umbrella, since we saw that in all cases the combination of a system, x, with MENE plus one other system, y, yielded an improvement in the performance of system x.

9.2 MENE's Portability to Other Languages

We believe that the results from the port of MENE to Japanese are also intriguing. That fact that the system achieved a result 6 points above the median score (albeit 6 points below the best score), in a language in which the author has no linguistic ability, would seem to indicate the MENE shows promise as a tool for languages of low diffusion (i.e. languages for which their are relatively few native-speaking computational linguists, and/or languages for which the amount of available online text is relatively small). Along these lines, it is encouraging to recall that the port of MENE from English to Japanese was accomplished in just 3 person-days (although we had a head-start in being able to use components of Sekine's Japanese named-entity decision-tree system [48], as discussed in section 8.1).

MENE's results on Japanese in the IREX evaluation were achieved on a 294,000-word corpus. While this is a relatively large corpus of tagged text, on the other hand, it is a relatively small corpus in an absolute sense. One could assume that if a language had such "low-diffusion" that there were not even 300,000 words of text available in electronic form, then nobody would have sufficient interest in the language to want to build an N.E. recognizer.

To port MENE to an arbitrary language, we see several issues:

- One needs 300,000 words of tagged text. As discussed above, securing 300,000 words of text should not be a problem. The tagging can be done in roughly 12 person-days (extrapolating figures from [36] [17]). Even assuming a doubling of the annotation time (some slow-down would be inevitable given that the tagging team for a new language would be unlikely to have any experience at

named entity tagging), this doesn't seem like a major expense given that the tagging can be done by native-speaking college students or college graduates with degrees in technical or non-technical fields [17].

- One needs a tokenizer. In English, we built this ourselves, while for the Japanese system we relied on the Juman lexical analyzer. In a language like Japanese which does not have white-space-separated words, MENE would have to rely on a tokenizer written by a native speaker. One might have a similar need for a sophisticated tokenizer in languages with complex morphologies, like German. In other languages, though, one could take two different approaches to the problem:
  - Work with a native speaker on a tokenizer. The tokenizer for the English MENE system is a set of relatively simple regular expressions written in Perl (i.e. separate the " 's" from the root word, quotation marks are not part of the word, etc.) One would expect that an educated native speaker (not even necessarily a linguist) could come up with a reasonable list of such rules.
  - Infer the tokenization rules from the training data. One can iteratively compare the named entities in the training data with the output of the current tokenizer to see how many names in the training data are not properly tokenized and adjust the tokenizer accordingly. Of course, one would expect that the annotators of the training data must have had a set of rules for determining name boundaries, which would seem to imply that the designer of the N.E. recognizer could simply work from these rules. However, testing a tokenizer against human-tagged data is always helpful, no matter what methodology one is using.
- For MENE to work in its current form, one needs a language in which words can be identified by local clues. In languages in which the subject or direct object of a verb is often widely separated from the verb (again, German might be an example here), we would lose the benefit of features which looked at the verb as a predictor of named entities. In order to circumvent this problem, we would need to add at least a minimal amount of parsing technology to the system.
- In a language which was more highly inflected than English, we might need to add a morphological analyzer to the system to derive root words from inflected words so that the root words could serve as lexical features. We speculate, though, that in an inflected language, one would want to generate three potential features from each inflected word: the root word, the inflec-

tion, and the complete inflected word. Each would supply a different level of specificity and, thus, each might be useful to the system.

While there are certain hurdles to overcome, we believe that the ability to port the system quickly and cheaply to a new language is another promising feature of the system.

9.3 Issues with Real-World Applications of the MENE System

As a stand-alone system, MENE has shown strong relative performance, while when combined with other named entity systems it has shown that it can achieve scores which in some cases are within the range of human performance. A logical question, then, is whether there are any barriers to transforming MENE from a research prototype into a practical system.

In our opinion, the training-time performance (2 hours 40 minutes when run on a 321,000-word training corpus on a 143 MHZ Sun Ultra-1) of the system should not be a barrier to its deployment in the field. Even assuming very fast annotators, the machine time will be at least an order of magnitude less than the time it would take to tag the training data.

As for ease of customization from a programmer's viewpoint, the fact that the system was easily ported from English to Japanese speaks for itself. In a very straightforward port to a new domain within the same language, where there are no issues with the formatting of the text, we think that the system can be ported in a day or two of programming time, with most of the time spent going towards error analysis and the possible addition of new features exposed by the errors. For instance, if one tests the system on a corpus in which dates are written as "YYYYMMDD", like "19980715", one might need a new pattern-feature like the ones discussed in section 5.1 so that the system will be able to tag these dates properly.

We think that the key barrier to a practical implementation of the system is the system's speed at run-time. As noted in section 4.4.4, the system is able to process about one megabyte of text per hour. This is comparable to the 6 MB/hr rate reported by BBN [5], but is over two orders of magnitude behind the speed of roughly 190 MB/hr which IsoQuest would have achieved on similar hardware.

One illustration of these comparative speeds is that the current system would take roughly 5 minutes to process this quarter-megabyte thesis on a 450 MHZ machine, while the IsoQuest system could accomplish the task in under 2 seconds. Although MENE's speed might be acceptable for some applications (like automatically building the index of a book as an add-on to a word-processing application),

it would be unacceptable for many others. In particular, the speed of the IsoQuest system is such that it could process information as it was being downloaded from the web without adding appreciably to the download time, while MENE is too slow for this sort of work.

However, we feel that MENE's current speed statistics are nowhere near the speed limits of a maximum entropy approach to named entity recognition. MENE was designed with flexibility and portability as the primary considerations. If the run-time system were rewritten with an eye for speed, we believe that one could obtain dramatic improvements. In particular, the current system uses a pre-processing stage which tokenizes the text, checks for the presence of various patterns (the "binary features" of section 5.1), looks up words in dictionaries, etc. and writes out the results of this pre-processing stage to disk. It seems likely that if this preprocessing were more tightly integrated with the rest of the code and we didn't have to worry about disk-IO, we could improve results dramatically.

Furthermore, we take a certain speed hit from our implementation of features as specializations of an abstract "Feature" base class, with feature firings being determined by the outcome of a virtual method which operates on abstract "History" and "Future" objects. A less generic implementation coded with speed as a primary objective could doubtless improve things considerably. Note that the Viterbi search phase of decoding (discussed in section 4.3) has the potential to be a bottleneck, but with an efficient implementation of a Viterbi search like the one used in MENE [50], the time required for the search phase is insignificant–at most a few percent of the total processing time.

It remains to be seen how fast an efficient maximum entropy named entity system would run, but one should note that a maximum entropy approach will always be at a certain disadvantage to an efficient handcoded system based on regular expressions. This is due to the requirement that the M.E. system perform the floating point arithmetic of equation 3.3 for each token, which if one assumed an average of 4.8 features per future (the average for the basic MENE system over our MUC-7 training corpus), would imply (for a 7-tag NE system) that we could compute the total number of floating point operations to be the following:

$$
\begin{aligned}
F &\equiv \text{Features per future} && (9.1)\\
T &\equiv \text{Number of NE tags} && (9.2)\\
U &\equiv \text{Number of Futures} = (F * T) + 1 && (9.3)\\
O &\equiv \text{Number of floating-point operations required per word} && (9.4)\\
&= \text{Denominator ops} + \text{divisions} && (9.5)\\
&= [(F-1) * U + (U-1)] + U && (9.6)\\
F &= 4.8 && (9.7)\\
T &= 7 && (9.8)
\end{aligned}
$$

$$O = 167.2 \tag{9.9}$$

We could reduce the number of operations by trying to carefully weed out some of the most commonly activating features, but nevertheless, it is clear that a maximum entropy system will always have to cope with a certain amount of computational overhead. However, some quick calculations indicate that this overhead is probably not that significant. A speed of 1 MB of text per hour implies a speed of approximately 20 miliseconds per token. If we made the conservative assumption that our 143 MHZ machine could perform 5 floating point operations per microsecond, then we are spending 33 microseconds on the floating-point math. These operations would amount to less than .2% of our run time, so it would appear unlikely that these calculations will prove to be the limiting factor in improving system speed.

9.4 Future Work

While work on improving the speed of the system is very important, further improvements in MENE's accuracy as a stand-alone system would also be very helpful. We feel that ultimately it is important to know whether a purely statistically trained system like MENE (or BBN's Identifinder system, for that matter), can achieve results which match the performance of the very best handcoded systems, like the one from IsoQuest.

9.4.1 Improving System Accuracy

We have a number of ideas for features which could be added to MENE to improve its accuracy:

1. Adding features which activate based on word prefixes and suffixes. These would be helpful for identifying words for which we had no lexical features. For instance, words with a suffix of "son" or a prefix of "Mc" would predict a person. Many different affixes indicate corporations, such as "Micro-", "-soft", "Iso-", and "-Quest". The challenge with adding these features would be to find a way to discover the affixes automatically. We might try Ratnaparkhi's technique from his maximum entropy part-of-speech tagger [40], where he incorporated these kinds of features by simply hypothesizing as features all affixes of length ≤ 4 which occurred in rare words (words with a count ≤ 4) found in his training corpus and then, in a second phase, selected those features which co-occurred with a given future more than 10 times.

2. Adding some form of parsing technology so that we would not be so reliant on predictive words occurring within a window running from $w_{-2} \ldots w_2$. For

instance, our system can identify the location "Kalamazoo" in the phrase "flew to Kalamazoo" much easier than in the phrase "flew with Mr. Jones to Kalamazoo". A correct parse of these phrases would allow us to add a feature which would predict "location" for a given word when its governing verb is "flew".

3. Adding better sentence-boundary detection such as the techniques reported in [41]. Currently we do not make any attempt to distinguish between a "." that ends a sentence and a "." which appears for some other reason. Doing this would allow us to better distinguish proper noun capitalization from sentence-beginning capitalization.

Another major avenue for improvement is in the related areas of detecting compound features and adding sophisticated feature selection techniques. We believe that we should get considerable gain from the inclusion of pattern-like compound features (i.e. w_{-2} = "flying" $\wedge$ w_{-1} = "to" predicts "location"). Since the relatively simple methods of detecting useful compound features which we described in section 6.2 failed to improve performance, we would have to look at some of the more sophisticated methods of detecting useful M.E. features in the literature, such as [4] [3] [38].

It would also be interesting to learn whether tighter integration with a hand-coded system like Proteus could result in higher scores for the hybrid system. In particular, the question would be whether we could have different features activating depending on which Proteus pattern was predicting a particular named entity, on the assumption that some Proteus patterns are more reliable than others. Our initial research along these lines was unsuccessful, but since this seems to be similar to the technique used by the LTG/Edinburgh system which achieved the highest score at MUC-7 [35], we think it is worth revisiting.

9.4.2 Harder Named Entity Tasks

The generally successful named entity results reported at the MUC-7 conference and the ending of DARPA's Tipster program which funded the evaluation have led the information extraction community to look for harder named entity recognition problems. Experimenting with these problems would be the next logical step forward for this research.

One way of increasing the difficulty of the task is to add more named entity categories. The addition of the artifact category in the IREX evaluation was one example of this, but one DARPA proposal for a named entity evaluation [13] included such new categories as "Animal", "Nationality", "Facility", "Award", "Financial" (i.e. a stock index), "Legal" (i.e. "The Gramm-Rudman amendment"), "Age",

"Temperature", "Identifier" (i.e. a social-security number), and many others for a total of 39 different named entity categories up from the current 7 categories. This proposal was ultimately scaled back to a proposed evaluation taking place this fall which will only add three categories: "Time duration", "Measurement", and "Cardinal" (a numeric count or quantity of some object) [12]. We expect that the addition of these three, relatively clear-cut categories will not greatly impact MENE's performance.

However, it is interesting to speculate on how MENE might have done on the cancelled 39-category named entity task. We only achieved 39.4% precision and 26.5% recall on the "artifact" category of the IREX evaluation, our lowest category score by far. On the other hand, artifacts are a particularly difficult category to define and, since the category is such a broad one, it is difficult for a system like MENE to pick up the contextual clues needed to identify these names. It could be that the more tightly focused categories like "Award" and "Legal" could be identified by a small number of lexical clues such as "prize" and "medal" for the former and "amendment" and "act" for the latter. More generally, expanding the number of futures which MENE needs to predict is likely to take a certain speed hit as it deals with the extra futures. Equation 9.6 shows that the growth will be no more than linear in the number of futures, and it could be considerably less than linear as many of these new fine-grained categories may have only a relatively small number of features activating on them.

The other way to increase the problem's difficulty is to work on text that is noisier than the news service text which was the corpus for the MUC-7 and IREX evaluations. The results reported on this at the 1999 DARPA Broadcast News Workshop were very positive, with BBN [36] reporting that their HMM-based system (essentially the same system discussed in sections 2.4 and 7.5) showed only a deterioration of 3.4 F-measures from moving from mixed-case, punctuated text to upper-case unpunctuated text (the format of speech transcripts) and then showed only a deterioration of 0.7 points of F-measure for every 1% increase in the word error-rate, which implies that not every speech error causes a named entity error. SRI reported similarly strong results with their hand-coded "TextPro" system [2].

As discussed in section 7.3, MENE's showed a greater degree of deterioration on upper-case text than BBN's Identifinder system. We have expended very little effort on improving MENE's performance in this area and suspect that it could be possible for it to do much better. On the other hand, the consensus coming out of the HUB-4 evaluation (for instance [2]) was that the results of the evaluation were so favorable that named entity recognition from speech was essentially a solved problem. It is not clear to us that we could do better than to equal the very strong results reported at the conference.

9.5 Summary

One way of defining the goal of the field of artificial intelligence (of which computational linguistics is a sub-specialty) is to say that we are working on problems which are easy for people but are hard for computers [19]. While this leaves out the category of problems which are difficult for both people and computers (like playing chess), it does include a vast range of problems which we take for granted but which are devilishly hard to program.

Named entity recognition is one small example of this sort of problem. N.E. represents a particularly tractable task from an AI standpoint, particularly for those who are attacking the problem with a statistical approach, due to the relative ease with which one can gather a large body of well-defined training data. We believe that it is a particularly suitable problem for maximum entropy because it is characterized by a large number of weakly predictive features which will almost always have certain exceptions. Taking the product of the features' weightings to see which future is predicted most strongly thus seems like a reasonable thing to do in this situation. Another factor working in our favor is that we only have to choose from a very limited range of tags (29 of them) for each word. A wider future space would have given us a much more computationally intensive problem and might have forced us to worry about limiting the feature pool and handling sparse data, which weren't major issues for us in this work.

As mentioned earlier, there are many examples of problems along these lines both inside and outside of computational linguistics. While M.E. has been used in physics for over 40 years, with the original work coming from that field [26], the first applications of M.E. modeling to artificial intelligence problems were very recent, with some of the first articles on it coming out in the 1994 - 1996 time period [44] [20] [4], all of it based on research which originated at IBM's T. J. Watson Research Center. These articles developed a method which allowed M.E. to be used to build conditional probabilistic models and led to this current work.

While maximum entropy models certainly benefit from finely crafted features which represent a deep understanding of the problem, our results on modeling for Japanese show that the technique is powerful enough that it can yield acceptable results with a set of features gathered by someone with only the most superficial understanding of the problem. On the other hand, the responsiveness of the English-language system to additional training data and additional sources of knowledge such as reference resolution and inputs from handcoded systems is an indication that the technique is able to squeeze a great deal of accuracy out of good information sources. In fact what we like most about maximum entropy modeling is that, to a large extent, it allows us to focus on adding to our knowledge of the problem, while the system takes care of the problem of integrating today's new insights with the ideas we added to the system yesterday.

Appendix A

Binary Partitioning Features in Maximum Entropy Models

A.1 Motivation

Good feature selection algorithms are critical to the success of a maximum entropy model. The process of selecting features for a maximum entropy model generally takes place in two distinct stages:

1. The modeler defines various classes of features which he/she believes are good predictors of the future given the available data from the history. For instance, in the MENE system we defined classes of features such as the following, which we described in chapter 5.

   (a) Lexical features: Does a particular word appear in a particular place in the $w_{-2} \ldots w_2$ window? (i.e. w_{-1} = "Mr" predicts the future "person_unique")

   (b) Dictionary features: Does w_0 appear on a particular word list (i.e. w_0 being on a list of common first names predicts the future "person_start")

   (c) Capitalization features: Features such as "w_0 capitalized predicts 'location_start' ".

2. From the set of all theoretically possible features in the selected classes, some automated process selects a subset based on the features' performance on training data.

"Subset" and "partitioning" features are the subject of this appendix. We say that $m(h, f)$ is a subset feature of $n(h, f)$ if for any $\langle h, f \rangle$ pair, $m(h, f) \Longrightarrow n(h, f)$, i.e. whenever m "fires", n also fires. Two features, q and r partition a third feature

s if $q(h,f) \Longrightarrow \overline{r(h,f)}$ and $r(h,f) \Longrightarrow \overline{q(h,f)}$ and $s(h,f) \Longrightarrow q(h,f) \vee r(h,f)$. The central result of section A.2 of this appendix is that a model containing the features q, r, and s is equivalent to a model containing any two of these three features. I.e. any one of these three features can be deleted without affecting the model, or, viewed another way, the modeler is free to decide which two of these three features are easiest to implement. Note that the same result would hold if s were partitioned by n features where $n \geq 1$. In the special case where $n = 1$, section A.2 shows that if $r(h,f) = s(h,f)$, then a model containing r and s would be equivalent to a model containing either r or s.

This result can be used in two different ways. First of all, it gives the modeler the freedom to choose which of the features q, r, or s he/she wishes to implement. We will show in section A.3 that we were able to use this result in several different ways in the MENE system. Secondly, the result can be used to automatically remove from the system features which are redundant and/or features which can be shown to have been trained on insufficient data. Hence the result is useful for feature selection by both machines and humans.

A.2 Some Proofs

Consider a maximum entropy model P_M which returns conditional probabilities $P_M(f|h)$ where f is a "future" in the domain F and h is a "history" in the domain H. Recall that conditional probabilities in a maximum entropy model are computed as in equation 3.3.

A.2.1 Theorem

Let us denote as G a set of binary features g_i which make up the model P_M. If b is a binary feature, let us define a caligraphic uppercase letter, $\mathcal{B}$, to represent the set $\langle h,f\rangle \in \langle H,F\rangle$ such that $b(h,f) = true$. Suppose we have a set of features $L = l_1 \ldots l_k(h,f)$ which meet the following condition:

$$\mathcal{L}_1 \cap \mathcal{L}_2 \ldots \cap \mathcal{L}_k = \emptyset \tag{A.1}$$

Let us define the feature m in terms of its subset over the domain $\langle H,F\rangle$ as:

$$\mathcal{M} = \cup_{i=1\ldots k}\mathcal{L}_i \tag{A.2}$$

Given a feature r such that

$$\mathcal{R} \supset \mathcal{M} \tag{A.3}$$

let us define the feature p as

$$\mathcal{P} = \overline{\mathcal{M}} \cap \mathcal{R} \tag{A.4}$$

Now, suppose that $L \subset G$ and $r \in G$. Then the addition of p to G will leave P_M unchanged.

A.2.2 Proof

The maximum entropy estimation procedure builds a model which must be in the form of equation 3.3 and in which the expected value of every feature g_i according to the M.E. model P_M equals the empirical expectation of g_i in the training corpus. This second requirement is expressed in the following equation (repeated here from equation 3.13).

$$\sum_{h,f} \tilde{P}(h,f) \cdot g_i(h,f) = \sum_{h} \tilde{P}(h) \cdot \sum_{f} P_M(f|h) \cdot g_i(h,f) \tag{A.5}$$

Here $\tilde{P}$ is an empirical probability. Of the space of all possible models $P(f|h)$, the intersection of the models which satisfies both of these conditions is unique and is the model which has maximum entropy subject to the constraints of equation A.5 [20]. Consequently, if we can show that after adding a feature, g, all constraints are satisfied, uniqueness guarantees that P_M is unaffected. In other words, in a model which contained the features $G \cup g$, g could be assigned the weighting $\alpha_g = 1$ in equation 3.3, so g would have no impact on P_M.

Since $r \in G$ and $L \subset G$, we have

$$\sum_{h,f} \tilde{P}(h,f) \cdot r(h,f) = \sum_{h} \tilde{P}(h) \cdot \sum_{f} P_M(f|h) \cdot r(h,f) \tag{A.6}$$

$$\forall_i \left[\sum_{h,f} \tilde{P}(h,f) \cdot l_i(h,f) = \sum_{h} \tilde{P}(h) \cdot \sum_{f} P_M(f|h) \cdot l_i(h,f) \right] \tag{A.7}$$

Summing A.7 over all i and then using equations A.1 and A.2, we get

$$\sum_{i} \left[\sum_{h,f} \tilde{P}(h,f) \cdot l_i(h,f) \right] = \sum_{i} \left[\sum_{h} \tilde{P}(h) \cdot \sum_{f} P_M(f|h) \cdot l_i(h,f) \right] \tag{A.8}$$

$$\sum_{h,f} \tilde{P}(h,f) \cdot m(h,f) = \sum_{h} \tilde{P}(h) \cdot \sum_{f} P_M(f|h) \cdot m(h,f) \tag{A.9}$$

Subtracting A.9 from A.6, we get

$$\sum_{h,f} \tilde{P}(h,f) \cdot (r(h,f) - m(h,f)) = \sum_{h} \tilde{P}(h) \cdot \sum_{f} P_M(f|h) \cdot (r(h,f) - m(h,f)) \tag{A.10}$$

Given that we are working with binary features, and from the definitions of p and r (in equations A.4 and A.3), we can write

$$p(h,f) = r(h,f) - m(h,f) \tag{A.11}$$

Substituting this into equation A.10, we get

$$\sum_{h,f} \tilde{P}(h,f) \cdot p(h,f) = \sum_{h} \tilde{P}(h) \cdot \sum_{f} P_M(f|h) \cdot p(h,f) \tag{A.12}$$

P_M satisfies the required constraint, so it doesn't change with the addition of $p(h,f)$.□

A.2.3 Corollary 1: Adding a "Partitioned Feature"

Let us assume all the conditions set out in section A.2.1 up through equation A.2. In addition, we are given a feature r such that

$$R = M \tag{A.13}$$

The set of features L partitions r, so we call r a "partitioned feature". Suppose that $L \subset G$. Then the addition of r to G will leave P_M unchanged.

[Note that taken together the theorem and this corollary means that if L partitions r, and $L \subset G$ and $r \in G$, then we can remove r or any $l_i \in L$ and P_M will remain unchanged.]

A.2.4 Proof of Corollary 1

Using equations A.7 through A.9, and then substituting $r(h,f)$ for $m(h,f)$ into equation A.9 (by this corollary's definition of r), we get

$$\sum_{h,f} \tilde{P}(h,f) \cdot r(h,f) = \sum_{h} \tilde{P}(h) \cdot \sum_{f} P_M(f|h) \cdot r(h,f) \tag{A.14}$$

□

A.2.5 Corollary 2: All but one future

Suppose that the future domain $F = \{f_1 \ldots f_k\}$. Define a set of features $S = \{s_1 \ldots s_k\} - s_j$, where $S \subset G$ and $s_j \notin G$ and

$$s_i(h,f) = \Gamma(h) \wedge (f = f_i) \tag{A.15}$$

where $\Gamma(h)$ is a binary function on the history domain. Then the addition of s_j to the model will leave P_M unaffected.

A.2.6 Proof of Corollary 2

The definition of the future domain F is that one element from F must be chosen for every history H. Therefore, for our arbitrary function $\Gamma(h)$, we could assume that there exists the following "phantom feature" in the model which imposes the constraint that one of the futures must be chosen:

$$w(h, f) = \Gamma(h) \wedge (f \in F) \tag{A.16}$$

Then identifying S as L and w as r from the theorem, it is clear that S forms a partition which fires on a subset of $\mathcal{W}$ and that the missing feature s_j satisfies the property that

$$\mathcal{S}_j = \overline{\mathcal{S}} \cap \mathcal{W} \tag{A.17}$$

Thus by the theorem, the addition of s_j leaves P_M unchanged. □

A.3 Applications

The most obvious application of this theorem is to apply corollary 2 to M.E. models which are only making binary predictions (i.e. choosing between the futures a and $\overline{a}$). Then the model should be built so that given two features whose history conditions are the same but whose future conditions are different, the feature is included which fires less often. The less common feature is chosen because training-time and run-time efficiency will be enhanced by having fewer active features in the model. This technique was used in a maximum entropy sentence parser [53] whose future space was {link,non-link} (whether two words were linked or not by a syntactic dependency). Since links are less common than non-links, only features predicting "link" were retained. This technique resulted in a model with roughly half the number of features and only a trivial loss of accuracy (the model had some cases where either link or non-link features were not present, so the small loss of accuracy could be attributed to cases where the non-link features were eliminated without the presence of a corresponding link feature).

Note that to this author's knowledge, the vast majority of maximum entropy systems have features which are in the form of equation A.15. These are known as "separable features" because they can be divided into a separate test of a history condition and a future condition. While Corollary 2 has limited use for a maximum entropy language model [3] where $|F|$ is very large, there are many instances of M.E. applications where the future space is quite small, such as the parser cited above, named entity recognition [8] ($|F| = 29$), part-of-speech tagging [40] ($|F| \approx 40$), or sentence-boundary detection [41] ($|F| = 2$). In these cases, it is a trivial matter to write a routine to scan the list of features included in the model, discover cases

in which there are features predicting every future in F, and then eliminate the feature from the list which is the most common.

Capitalization issues proved to be another example in which this theorem was useful for the named entity system. As we wrote in section 5.1, in our system we had a feature, f_1, predicting person_unique given that the current word began with a capital letter. We also had a feature f_2 which predicted the same future given that the current word was all capital letters. Clearly $\mathcal{F}_2 \subset \mathcal{F}_1$. In a non-M.E. system such as [5], the approach taken to this situation is to have the features that are included be (using our set notation again) $\mathcal{F}_2$ and $\mathcal{F}_1 \cap \overline{\mathcal{F}_2}$ (an all-caps feature and an init-cap-but-not-all-caps feature). Clearly these latter two features partition the space of the init-cap feature, so by our theorem we will get the same results if we include any two of the three. We chose to include the init-cap and all-cap features because they were the simplest to implement.

Another area where these results were useful was with our "dictionary features". As discussed in section 5.4, we were able to use the freedom granted to us by the theorems of this appendix to avoid the necessity of "fine-tuning" our dictionaries to weed out potentially "dangerous" entries, like the presence of the word "April" in a first-name dictionary. As we argued in that section, the first-name dictionary feature will be a superset feature relative to a "subset" lexical feature for "April". Hence, it is irrelevant to system performance whether or not we edit "April" out of our dictionary (although note the caveats discussed in section 5.4–if "April" occurs just once or twice as a person-name, the presence/absence of "April" in the dictionary may have a minor impact).

The final point to be made here is that when a set of features, L, defines a space $\mathcal{L}$ which is a subset of the space $\mathcal{R}$ for a feature r, we have the option of replacing r with a feature defining the space $\mathcal{R} \cap \overline{\mathcal{L}}$. While this is the reverse of the decision we made for the capitalization features, this kind of switch can speed convergence of the system during training, as discussed in section A.3.1.

A.3.1 Convergence Issues

The issue of convergence prevents us from being completely free to apply the results of the proof. While it is theoretically true that at the end of training, the α weighting of every feature will be set so as to satisfy equation A.5, (and consequently all the results in section A.2 will hold), in reality feature expectations converge gradually towards these points. As has been noted before [44], the number of iterations required for convergence is roughly proportional to the number of active features for each history, although the IIS algorithm [20] improves matters somewhat over the GIS algorithm.

If one is concerned about achieving rapid convergence, then one should use the freedom granted to the modeler by the proof to eliminate features which are

partitioned completely by subset features. Furthermore, the modeler can replace superset features with features which do not activate on the $\langle H, F \rangle$ space defined by the largest set of their subset features which meets the conditions of equation A.1. This will reduce the number of simultaneously active features and, as argued above, this model will be theoretically equivalent, but will converge faster than the model with the original superset feature. The results shown in section A.5 bore out this claim.

A.3.2 Training vs. Testing

The theorem shows that if the features r and L are all in a model G and L partitions r, then one of these features is redundant, which is another way of saying that one of them could be assigned an α weighting of unity. However, it doesn't say that the redundant feature's weight must be unity. In fact, the weighting of the redundant feature is a free parameter and all the other features are dependent parameters. To see this, consider a model, P_M in which $L \in G, r \notin G$ which assigns the weightings $\alpha_{l_1}, \alpha_{l_2} \ldots \alpha_{l_k}$ to $l_1 \ldots l_k$. Now consider G', which defines the model P'_M where $L \in G, r \in G$. Since we must have $\forall_{h,f} P_M(f|h) = P'_M(f|h)$, given an arbitrary α_r arrived at by the estimation procedure, we must have $\alpha'_{l_i} \cdot \alpha_r = \alpha_{l_i}$. Consequently, we will have $\alpha'_{l_i} = \frac{\alpha_{l_i}}{\alpha_r}$.

This is all of merely theoretical interest in models in which the relevant partitioning conditions necessarily hold over all data, such as was the case with the init-cap and all-cap features. However, we will see this behavior whenever the partitioning behavior described in the theorem holds over the histories in the training corpus, even if a test corpus could turn out otherwise. From the argument above, it is clear that we will have arbitrary and unpredictable results on any feature, r which is partitioned over the training corpus by a set of features L if for a given $\langle h, f \rangle$ in the training corpus, $r(h, f)$ activates and no $l_i(h, f)$ activates or if an $l_i(h, f)$ activates without $r(h, f)$. We describe some experiments in section A.5 in which we eliminated features which were completely or almost completely partitioned by their subset features.

A.3.3 Sparse Data

Many M.E. models impose a minimum count on the number of times a feature must occur in the training corpus before it is included in a model since the statistics on rarely occurring features are likely to be inaccurate. For instance, in this work, we generally used a threshold of three occurrences. In [40] a cutoff of 10 was used. This theorem shows, however, that such raw counts may not be indicative of the number of history events which are being used to determine the feature's weighting. For instance, given features r and s, where $\mathcal{S} \subset \mathcal{R}$ and $|\mathcal{S}| = 10$ and $|\mathcal{R}| = 11$,

although r appears to be trained on an adequate dataset, in reality this model is equivalent to a model which contains only s and $r - s$, where the latter feature was trained on only one event. Such a model may prove inaccurate in situations where r activates without s. This hypothesis was also tested in experiments described in section A.5.

A.4 The NP-Completeness of Detecting Maximum Partitioning Subsets

While detecting cases in which one feature is a subset of another is a trivial task, there is a problem with determining the largest set of partitioning features from among the features which are a subset of a feature r. Doing this is necessary for all of the experiments in the automatic modification of the feature pool discussed in section A.5. Given a feature r and a set of features C where every feature $c_i \in C$ is a subset feature of r, we wish to detect the subset L which maximizes $|\mathcal{L}|$ (the number of $\langle h, f \rangle$ events on which any feature in L fires) and for which equation A.1 holds (no two features in L fire on the same $\langle h, f \rangle$ pair). This problem can be represented as a graph in which every $l_i \in L$ is a node which is labeled with a weight representing $|\mathcal{L}_i|$ and we draw an edge between l_i and l_j if the two features co-occurred in the training corpus. The problem is to find the set of vertices, L, which has the highest possible sum of weight labels, "L-weight", and no two vertices are connected by an edge. Formally, we wish to determine if there is a set of vertices whose L-weight $> K$. K is an arbitrary constant. Let's call this problem "Max-Weighted-No-Edge".

Unfortunately, Max-Weighted-No-Edge is NP-complete. We can prove this by observing, first, that the problem is clearly in P because we can guess a set of vertices, verify that no two vertices share an edge, and verify that the sum of the weights is $> K$. We can show that the problem is NP-hard by a reduction from the "Clique" problem. Clique is known to be NP-complete [15] and is the problem of finding a subset of vertices V' in a graph, $G = (V, E)$, where each pair of vertices in V' is connected by an edge and $|V'| = K$ (informally, we are looking for the largest possible $|V'|$).

We will reduce Clique to Max-No-Edge, the special case of Max-Weighted-No-Edge in which every vertex has a weight of one. Clearly if Max-No-Edge is NP-hard, then so is Max-Weighted-No-Edge. Clique can be reduced to Max-No-Edge by computing the "complement" of G, $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(u, v) : (u, v) \notin E\}$. It is clear from this reduction that there is a set of non-edge-sharing vertices, L, in $\overline{G}$ where $|L| = K$ if and only if G has a clique of size K.

Given this result, we simply limited our search for partitioning features to the K subset features with the largest count for each "superset" feature, where we used

$K = 16$. Because of the NP-completeness, we basically used a naive try-all-pairs approach for our search.

A.5 Results

We ran a series of experiments which sought to test the effects of three sorts of modifications to our feature pool based on the theoretical results of this appendix:

1. Removing any feature, r, which is completely partitioned by a set of "subset" features, L over the training corpus.

2. Removing any feature, r, which has a set of non-overlapping subset features, L, which account for all but K or fewer of the number of activations of r on the training corpus, as advocated in section A.3.3. We used $K = 2$ in our experiments.

3. Given a feature, r, whose largest set of non-overlapping subset features is L, remove r and replace it with a feature which fires on the $\langle h, f \rangle$ space $\mathcal{R} - \mathcal{L}$ in order to speed convergence, as suggested in section A.3.1.

Our experiments focused on testing the system on runs using our basic MUC-7 MENE-only system when trained on the 100 articles of the official training corpus. Table A.1 below summarizes the test of combinations of these three different experiments which we tested with different numbers of iterations of the maximum entropy training algorithm in order to examine the models' convergence properties.

	Number of Max. Ent. Training Iterations						
System	50	100	150	200	250	300	350
Baseline	86.43	88.26	88.79	89.17	89.23	89.55	89.57
Experiment 1	86.44	88.31	88.84	89.08	89.18	89.51	89.57
Experiments 1, 2	86.53	88.14	88.78	89.11	89.23	89.39	89.40
Experiments 1,2, 3	86.89	88.42	89.00	89.34	89.24	89.31	89.31
Experiments 1, 3	86.94	88.35	89.07	89.23	89.28	89.46	89.53

Table A.1: Results from running the experimental models described on page 91 against dry run test data

The significant positive result that we can show from this table is that using the complemented (experiment 3) features does speed convergence. At 50 iterations, the 1-2-3 model performed better than the 1-2 model by 0.36 F and the 1-3 model outperformed the "1" model by 0.5 F. This advantage disappeared as we trained

the model further, but that is consistent with our theoretical predictions, as we discuss further in section A.6. At 350 training iterations, for instance, our scores on the two models with complemented features had moved from slightly higher to slightly lower.

In looking at these results, we think that it is important to note that although the features which are superset features tend to be the more frequently firing ones, the number of features impacted by the automatic methods described in this chapter was relatively small. Table A.2 shows the number of features affected by the three different experiments.

System Description	Experiment Number	Number of Features Affected
Total number of features in baseline system	Baseline	9242
Delete superset features with modified count of zero	1	55
Delete superset features with modified count of 1 or 2	2	43
Replace superset features with complemented features	3	380

Table A.2: Number of features affected by the experimental models from page 91

A.6 Conclusion

We feel that the results of section A.5 support the basic conclusions of this appendix. On the one hand, replacing superset features with features which exclude their subset features ("complemented features") clearly helps the model to converge faster, as predicted in section A.3.1. On the other hand, if one has the patience to let the model converge more completely, then one can simply use the original superset features because we show that with 350 iterations, any performance advantage of the complemented features disappears. This dovetails nicely with the prediction of section A.2 that, when the model has converged, complemented features and superset features will perform identically.

The results for experiment 1 (removing superset features completely covered by subset features from the feature pool) were that the system converged slightly faster, but ended up with the same performance given enough training iterations, which essentially agrees with our theoretical predictions. Experiment 2 (removing superset features with only 1 or 2 occurrences uncovered by subset features, as

advocated in section A.3.3) was somewhat less compelling. Convergence improved at very low numbers of iterations, but results were mildly negative relative to the baseline model when the system had converged. We posit two possible explanations for the failure of the system to improve after convergence:

1. The number of features affected by the experiment may have been too low for it to show much of an effect, as shown in table A.2.

2. We argued in section A.3.3 that these features should be eliminated because they were trained on sparse data. However, the results of training on sparse data are highly dependent on luck. Sometimes the data is representative and sometimes it is not. In the baseline model, we may have been lucky that the one or two examples on which these superset features were trained were representative of the data we encountered in the dry run test corpus.

However, we feel that the results of this appendix are likely to be most useful for "human feature selection". We found that the freedom which we gained by being able to decide whether to use a superset feature or a complemented feature was quite important to us by enabling us to build our features in the simplest possible way. The results of section A.5 show that a decision on whether or not to use complemented features has very little impact on system performance when the model has throughly converged. Hence, if, as was the case in our system, the marginal cost of training the system for extra iterations of the maximum entropy estimation algorithm is not too great, the modeler is free to ignore these questions and do what seems easiest and most intuitive.

Appendix B

MENE Output on MUC-7 Official Walk-Through Article

Below is the official "walk-through" article from MUC-7 along with the output of our best MENE-only results on the article (the "MENE-only with reference resolution" system trained on rotated data described in section 7.4 which scored an F-measure of 86.56 on the formal run). Our results on this article were fairly typical of our overall results as we achieved an F-measure of 85.50 with a precision of 88 and a recall of 84.

B.1 The Walk-Through Article

```
<DOC>
<DOCID> nyt960214.0704 </DOCID>
<STORYID cat=f pri=u> A4479 </STORYID>
<SLUG fv=taf-z> BC-MURDOCH-SATELLITE-NYT </SLUG>
<DATE> 02-14 </DATE>
<NWORDS> 0608 </NWORDS>
<PREAMBLE>
BC-MURDOCH-SATELLITE-NYT
MURDOCH SATELLITE FOR LATIN PROGRAMMING EXPLODES ON TAKEOFF
(kd)
By MARK LANDLER
c.1996 N.Y. Times News Service
</PREAMBLE>
<TEXT>
<p>
   A Chinese rocket carrying a television satellite exploded
seconds after launch Wednesday, dealing a potential blow to Rupert
```

Murdoch's ambitions to offer satellite programming in Latin America.
<p>
Murdoch's News Corp. is one of four media companies in a partnership that had leased space on the Intelsat satellite to offer the Latin American service. The other partners are Tele-Communications Inc., the nation's largest cable operator; Grupo Televisa SA, the Mexican broadcaster and publisher, and the giant Brazilian media conglomerate Globo.
<p>
Llennel Evangelista, a spokesman for Intelsat, a global satellite consortium based in Washington, said the accident occurred at 2 p.m. EST Wednesday, or early Thursday morning at the Xichang launch site in Sichuan Province in southwestern China. ``We have no details on what caused the accident,'' he said.
<p>
Evangelista said the Chinese-built Long March rocket veered off course and was destroyed after it failed to reach orbit. Intelsat was using the Long March rocket for the first time to launch one of its satellites. Intelsat currently has 23 satellites in orbit.
<p>
A spokesman for News Corp., Howard Rubenstein, said the accident would not hinder the group's plans to offer 150 channels of entertainment, news and sports programming to viewers in Latin America and the Caribbean.
<p>
``News Corp. has a number of other real options and will disclose them shortly,'' Rubinstein said in a statement.
<p>
Grupo Televisa and Globo plan to offer national and local programming in Spanish and Portuguese. Initially, the venture's partners said they planned to invest $500 million.
<p>
But a similar explosion last year delayed the plans of several American media companies to offer a package of satellite television services in Asia. Viacom, Time Warner's Home Box Office and Turner Broadcasting System were among the companies that had leased space on an Apstar 2 satellite to beam MTV, CNN and other channels throughout Asia.
<p>
After the rocket carrying that satellite exploded, media analysts said the companies had to settle for space on a series of regional satellites, which had less reach than the Apstar 2 would

have offered.
<p>
   News Corp. actually benefited from that accident. In 1993, the company had purchased a controlling stake in a rival Asian satellite service, Star TV. With his biggest competitors unable to enter the Asian market, Murdoch was able to build Star TV into the dominant programming service.
<p>
   A spokeswoman for Tele-Communications, LaRae Marsik, said the partners in the Latin American venture intended to begin service by the end of 1996. When the companies announced their plans last November, they said they planned to be in business by May.
<p>
   Ms. Marsik said Tele-Communications and its partners had a back-up plan, which could include leasing space on another satellite, but she declined to offer details. ‘‘It is an unfortunate incident,’’ she said, ‘‘but it is not a make-it-or-break-it event for us.’’
<p>
   Jessica Reif, a media analyst at Merrill Lynch & Co., said, ‘‘If they can get up and running with exclusive programming within six months, it doesn’t set the venture back that far.’’
<p>
   Hughes Electronics, a subsidiary of the General Motors Corp., is starting its own satellite broadcast service in Latin America. Ms. Reif said that venture, which is based on Hughes’s DirecTV service in the United States, would benefit if the explosion delayed the Murdoch-led venture.
</TEXT>
<TRAILER>
NYT-02-14-96 2029EST
</TRAILER>
</DOC>

B.2 MENE’s Annotations of the Walk-Through Article

<DOC>
<DOCID> nyt960214.0704 </DOCID>
<STORYID cat=f pri=u> A4479 </STORYID>
<SLUG fv=taf-z> BC-<ENAMEX TYPE="PERSON">MURDOCH</ENAMEX>-SATELLITE-NYT </SLUG>
<DATE> <TIMEX TYPE="DATE">02-14</TIMEX> </DATE>
<NWORDS> 0608 </NWORDS>

<PREAMBLE>
BC-<ENAMEX TYPE="PERSON">MURDOCH</ENAMEX>-SATELLITE-NYT
<ENAMEX TYPE="PERSON">MURDOCH</ENAMEX> SATELLITE FOR LATIN PROGRAMMING EXPLODES ON TAKEOFF
(kd)
By <ENAMEX TYPE="PERSON">MARK LANDLER</ENAMEX>
c.<TIMEX TYPE="DATE">1996</TIMEX> <ENAMEX TYPE="ORGANIZATION">N.Y. Times News Service</ENAMEX>
</PREAMBLE>
<TEXT>
<p>
A Chinese rocket carrying a television satellite exploded seconds after launch <TIMEX TYPE="DATE">Wednesday</TIMEX>, dealing a potential blow to <ENAMEX TYPE="PERSON">Rupert Murdoch</ENAMEX>'s ambitions to offer satellite programming in <ENAMEX TYPE="LOCATION">Latin America</ENAMEX>.
<p>
<ENAMEX TYPE="PERSON">Murdoch</ENAMEX>'s <ENAMEX TYPE="ORGANIZATION">News Corp.</ENAMEX> is one of four media companies in a partnership that had leased space on the <ENAMEX TYPE="ORGANIZATION">Intelsat</ENAMEX> satellite to offer the Latin American service. The other partners are <ENAMEX TYPE="ORGANIZATION">Tele-Communications Inc.</ENAMEX>, the nation's largest cable operator; <ENAMEX TYPE="ORGANIZATION">Grupo Televisa SA</ENAMEX>, the Mexican broadcaster and publisher, and the giant Brazilian media conglomerate Globo.
<p>
<ENAMEX TYPE="PERSON">Llennel Evangelista</ENAMEX>, a spokesman for <ENAMEX TYPE="ORGANIZATION">Intelsat</ENAMEX>, a global satellite consortium based in <ENAMEX TYPE="LOCATION">Washington</ENAMEX>, said the accident occurred at <TIMEX TYPE="TIME">2 p.m. EST</TIMEX> <TIMEX TYPE="DATE">Wednesday</TIMEX>, or early <TIMEX TYPE="DATE">Thursday</TIMEX> <TIMEX TYPE="TIME">morning</TIMEX> at the Xichang launch site in <ENAMEX TYPE="LOCATION">Sichuan Province</ENAMEX> in southwestern <ENAMEX TYPE="LOCATION">China</ENAMEX>. ``We have no details on what caused the accident,'' he said.
<p>
<ENAMEX TYPE="PERSON">Evangelista</ENAMEX> said the Chinese-built Long <TIMEX TYPE="DATE">March</TIMEX> rocket veered off course and was destroyed after it failed to reach orbit. Intelsat was using the <ENAMEX TYPE="LOCATION">Long March</ENAMEX> rocket for the first time to launch one of its satellites. Intelsat currently has 23 satellites in orbit.
<p>
A spokesman for <ENAMEX TYPE="ORGANIZATION">News Corp.</ENAMEX>, <ENAMEX TYPE="PERSON">Howard Rubenstein</ENAMEX>, said the accident would not hinder the group's plans to offer 150 channels of entertainment, news and sports programming to viewers in <ENAMEX TYPE="LOCATION">Latin America</ENAMEX> and the <ENAMEX TYPE="LOCATION">Caribbean</ENAMEX>.
<p>
``<ENAMEX TYPE="ORGANIZATION">News Corp.</ENAMEX> has a number of other real options and will disclose them shortly,'' <ENAMEX TYPE="PERSON">Rubinstein</ENAMEX> said in a statement.

<p>
<ENAMEX TYPE="ORGANIZATION">Grupo Televisa</ENAMEX> and Globo plan to offer national and local programming in Spanish and Portuguese. Initially, the venture's partners said they planned to invest <NUMEX TYPE="MONEY">$500 million</NUMEX>.
<p>
But a similar explosion <TIMEX TYPE="DATE">last year</TIMEX> delayed the plans of several American media companies to offer a package of satellite television services in <ENAMEX TYPE="LOCATION">Asia</ENAMEX>. Viacom, <ENAMEX TYPE="ORGANIZATION">Time Warner</ENAMEX>'s Home Box Office and <ENAMEX TYPE="ORGANIZATION">Turner Broadcasting</ENAMEX> System were among the companies that had leased space on an Apstar 2 satellite to beam <ENAMEX TYPE="ORGANIZATION">MTV</ENAMEX>, CNN and other channels throughout <ENAMEX TYPE="LOCATION">Asia</ENAMEX>.
<p>
After the rocket carrying that satellite exploded, media analysts said the companies had to settle for space on a series of regional satellites, which had less reach than the Apstar 2 would have offered.
<p>
<ENAMEX TYPE="ORGANIZATION">News Corp.</ENAMEX> actually benefited from that accident. In <TIMEX TYPE="DATE">1993</TIMEX>, the company had purchased a controlling stake in a rival Asian satellite service, <ENAMEX TYPE="LOCATION">Star</ENAMEX> TV. With his biggest competitors unable to enter the <ENAMEX TYPE="LOCATION">Asian</ENAMEX> market, <ENAMEX TYPE="PERSON">Murdoch</ENAMEX> was able to build <ENAMEX TYPE="LOCATION">Star</ENAMEX> TV into the dominant programming service.
<p>
A spokeswoman for <ENAMEX TYPE="ORGANIZATION">Tele-Communications</ENAMEX>, <ENAMEX TYPE="PERSON">LaRae Marsik</ENAMEX>, said the partners in the Latin American venture intended to begin service by the <TIMEX TYPE="DATE">end of 1996</TIMEX>. When the companies announced their plans <TIMEX TYPE="DATE">last November</TIMEX>, they said they planned to be in business by <TIMEX TYPE="DATE">May</TIMEX>.
<p>
Ms. <ENAMEX TYPE="PERSON">Marsik</ENAMEX> said <ENAMEX TYPE="ORGANIZATION">Tele-Communications</ENAMEX> and its partners had a back-up plan, which could include leasing space on another satellite, but she declined to offer details. ``It is an unfortunate incident,'' she said, ``but it is not a make-it-or-break-it event for us.''
<p>
<ENAMEX TYPE="PERSON">Jessica Reif</ENAMEX>, a media analyst at <ENAMEX TYPE="ORGANIZATION">Merrill Lynch & Co.</ENAMEX>, said, ``If they can get up and running with exclusive programming within six months, it doesn't set the venture back that far.''
<p>
Hughes Electronics, a subsidiary of the <ENAMEX TYPE="ORGANIZATION">General Motors Corp.</ENAMEX>, is starting its own satellite broadcast service in <ENAMEX TYPE="LOCATION">Latin America</ENAMEX>. Ms. <ENAMEX TYPE="PERSON">Reif</ENAMEX> said that venture, which is based on Hughes's

```
<ENAMEX TYPE="ORGANIZATION">DirecTV</ENAMEX> service in the <ENAMEX
TYPE="LOCATION">United States</ENAMEX>, would benefit if the explosion delayed
the <ENAMEX TYPE="PERSON">Murdoch</ENAMEX>-led venture.
</TEXT>
<TRAILER>
NYT-<TIMEX TYPE="DATE">02-14-96</TIMEX> <TIMEX TYPE="TIME">2029EST</TIMEX>
</TRAILER>
</DOC>
```

B.3 MENE's Errors on the Walk-Through Article

Below are the errors made by the system on this article. The column "Key-type" is the correct named-entity tag given to the phrase "Resp-type" is MENE's response, i.e. the tag which MENE assigned. Similarly "Key-text" and "Resp-text" are the correct text and MENE's text for the named entity. Note that in the first two errors, MENE assigned the right tag but tagged only an incomplete portion of the entity. The next 10 errors were cases where MENE omitted a tag which should have been there. The errors marked "spu" were spurious tags assigned by MENE. The last two were optional tags. MENE was neither penalized nor rewarded for missing them.

Note that MENE split the time entity "early Thursday morning" into the time "morning" and the date "Thursday". It got partial credit for the former and was penalized for a spurious tag on the latter. Note also that MENE mis-labeled the "Long March" rocket twice, once by identifying "March" as a date, and the second time by identifying "Long March" as a location. The latter tagging is puzzling until one considers that "Long Island" is a location, so the system had a moderately weighted feature predicting "location_start" given w_0 = "long". When we consider this along with the fact that the feature predicting "date" given "march" wasn't given a particularly high weight since "march" can appear in other contexts, we can see that the system was somewhat ambivalent on how to tag "Long March". Both of its responses were wrong, but both were reasonable.

```
TYPE TEXT KEY_TYPE   RESP_TYPE  KEY_TEXT                      RESP_TEXT

cor  inc  ORG.       ORG.       "Turner Broadcasting System" "Turner Broadcasting"
cor  inc  TIME       TIME       "early Thursday morning"     "morning"
mis  mis  ORG.                  "Globo"                      ""
mis  mis  LOCATION              "Xichang"                    ""
mis  mis  ORG.                  "Intelsat"                   ""
mis  mis  ORG.                  "Intelsat"                   ""
mis  mis  ORG.                  "Globo"                      ""
mis  mis  ORG.                  "Viacom"                     ""
```

mis	mis	ORG.		"Home Box Office"	""
mis	mis	ORG.		"Hughes Electronics"	""
mis	mis	ORG.		"Hughes"	""
mis	mis	DATE		"within six months"	""
spu	spu		LOCATION	""	"Long March"
spu	spu		ORG.	""	"MTV"
spu	spu		LOCATION	""	"Star"
spu	spu		LOCATION	""	"Asian"
spu	spu		LOCATION	""	"Star"
spu	spu		DATE	""	"Thursday"
spu	spu		DATE	""	"March"
opt	opt	ORG.		"Star TV"	""
opt	opt	ORG.		"Star TV"	""

Bibliography

[1] ANSI/ISO C++ standard, November 1997. Electronic version of the standard available at http://www.ncits.org.

[2] Appelt, D. E., and Martin, D. Named entity extraction from speech: Approach and results using the TextPro system. In *Proceedings of the DARPA Broadcast News Workshop (HUB-4)* (February 1999).

[3] Berger, A., and Printz, H. A comparison of criteria for maximum entropy/minimum divergence feature selection. In *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing* (June 1998), N. Ide and A. Boutilainen, Eds., The Association for Computational Linguistics, pp. 97–106.

[4] Berger, A. L., Della Pietra, S. A., and Della Pietra, V. J. A maximum entropy approach to natural language processing. *Computational Linguistics 22*, 1 (1996), 39–71.

[5] Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. Nymble: a high-performance learning name-finder. In *Fifth Conference on Applied Natural Language Processing* (1997).

[6] Black, W. J., Rinaldi, F., and Mowatt, D. Facile: Description of the NE system used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[7] Borthwick, A., Florian, R., and Papineni, K., 1997. MENE's history, future, and feature base classes were influenced by the architecture of a maximum entropy language model jointly developed by these three authors at IBM Watson Labs, Yorktown Heights, New York, in the summer of 1997.

[8] Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proceedings of the Sixth Workshop on Very Large Corpora* (August 1998).

[9] Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. NYU: Description of the MENE named entity system as used in MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[10] Brown, P., Della Pietra, S., Della Pietra, V., Mercer, R., Nadas, A., and Roukos, S. A maximum penalized entropy construction of conditional log-linear language and translation models using learned features and a generalized Csiszar algorithm. Unpublished IBM research report.

[11] Chinchor, N. MUC-7 scoring methodology. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (April 1998).

[12] Chinchor, N., Brown, E., Ferro, L., and Robinson, P. 1999 named entity recognition task definition, version 1.1 [draft], July 1999. Draft proposal for the 1999 HUB-4 named entity task.

[13] Chinchor, N., Brown, E., and Robinson, P. Event99 named entity task definition, version 5.0 [draft], April 1999. Draft proposal for named entity task. This proposal was later substantially revised.

[14] Chinchor, N., and Marsh, E. MUC-7 named entity task definition. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998). Available at http://www.muc.saic.com/.

[15] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. *Introduction to Algorithms*. MIT Press, 1990, ch. 36.

[16] Cover, T. M., and Thomas, J. A. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, 1991, ch. 11.

[17] Crystal, M. R., and Kubala, F. Studies in data annotation effectiveness. In *Proceedings of the DARPA Broadcast News Workshop (HUB-4)* (February 1999), vol. 1.

[18] Darroch, J., and Ratcliff, D. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics 43* (1972), 1470–1480.

[19] Davis, E. Lecture for Introduction to Artificial Intelligence Class, New York University.

[20] Della Pietra, S., Della Pietra, V., and Lafferty, J. Inducing features of random fields. Tech. Rep. CMU-CS-95-144, Carnegie Mellon University, 1995.

[21] Eriguchi, Y., and Kitani, Y. NTT Data: Description of the Erie system used for MET-2 Japanese. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[22] Fukumoto, J., Masumi, F., Shimohata, M., and Sasaki, M. Oki Electric Industry: Description of the Oki system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (April 1998). Available at http://www.muc.saic.com/.

[23] Fukumoto, J., and Shimohata, M., 1999. Personal communication from J. Fukumoto and M. Shimohata.

[24] Good, I. J. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *Annals of Mathematical Statistics 34* (1963), 911–934.

[25] Grishman, R. The NYU system for MUC-6 or where's the syntax? In *Proceedings of the Sixth Message Understanding Conference* (November 1995), Morgan Kaufmann.

[26] Jaynes, E. T. Information theory and statistical mechanics. *Physics Reviews 106* (1957), 620–630.

[27] Jaynes, E. T. Notes on present status and future prospects. In *Maximum Entropy and Bayesian Methods* (1990), W. T. Grandy and L. H. Schick, Eds., Kluwer, pp. 1–13.

[28] Jaynes, E. T. Probability theory: The logic of science. Manuscript for book. Available at http://bayes.wustl.edu/etj/prob.html, July 1998.

[29] Krupka, G. R., and Hausman, K. IsoQuest: Description of the NetOwl(tm) extractor system as used in MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[30] Lin, D. Using collocation statistics in information extraction. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[31] Magerman, D. M. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University, 1994.

[32] Marsh, E., and Perzanowski, D. MUC-7 evaluation of I.E. technology: Overview of results. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[33] Masui, F., 1998. Personal communication from Fumito Masui.

[34] Matumoto, Y., Kurohashi, S., Yamaji, O., Taeki, Y., and Nagao, M. Japanese morphological analyzing system: Juman. Kyoto University and Nara Institute of Science and Technology, 1997.

[35] Mikheev, A., and Grover, C. LTG: Description of the NE recognition system used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (April 1998).

[36] Miller, D., Schwartz, R., Weischedel, R., and Stone, R. Named entity extraction from broadcast news. In *DARPA Broadcast News Workshop (HUB-4)* (February 1999).

[37] Miller, S., Crystal, M., Fox, H., Ramshaw, L., Schwartz, R., Stone, R., Weischedel, R., and the Annotation Group. Algorithms that learn to extract information–BBN: Description of the SIFT system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (April 1998).

[38] Printz, H. Fast computation of maximum entropy/minimum divergence model feature gain. In *Proceedings of the Fifth International Conference on Spoken Language Processing* (November 1998).

[39] Quinlan, R. J. *C4.5: Program for Machine Learning*. Morgan Kaufman Publishers, 1993.

[40] Ratnaparkhi, A. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing* (May 1996), University of Pennsylvania, pp. 133–142.

[41] Reynar, J. C., and Ratnaparkhi, A. A maximum entropy approach to identifying sentence boundaries. In *Fifth Conference on Applied Natural Language Processing* (April 1997), pp. 16–19.

[42] Ristad, E. S. Maximum entropy modeling for discrete domains. Tutorial given at the M3D Workshop, University of Illinois at Chicago.

[43] Ristad, E. S. Maximum entropy modeling toolkit, release 1.6 beta, February 1998. Includes documentation which has an overview of MaxEnt modeling.

[44] Rosenfeld, R. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, 1994. CMU Technical Report CMU-CS-94-138.

[45] Sekine, S., 1998. Personal communication from Satoshi Sekine.

[46] Sekine, S. NYU system for Japanese NE - MET2. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[47] Sekine, S., 1999. Personal communication from Satoshi Sekine, one of the organizers of IREX.

[48] Sekine, S., Grishman, R., and Shinnou, H. A decision tree method for finding and classifying names in Japanese texts. In *Proceedings of the Sixth Workshop on Very Large Corpora* (1998).

[49] Sterling, J., 1998. Personal communication from John Sterling.

[50] Straszheim, T., 1998. The Viterbi search routine used in this work was written by Troy Straszheim.

[51] Stroustrup, B. *The C++ Programming Language, Third Edition.* Addison-Wesley, 1997.

[52] The egcs project. egcs 1.1. http://egcs.cygnus.com, September 1998. egcs is a free compiler for ANSI/ISO C++, C, Fortran, and other languages.

[53] Uchimoto, K., and Sekine, S. Japanese dependency structure analysis based on maximum entropy models. In *Proceedings of the European Association for Computational Linguistics* (1999). Accepted for EACL 1999.