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.

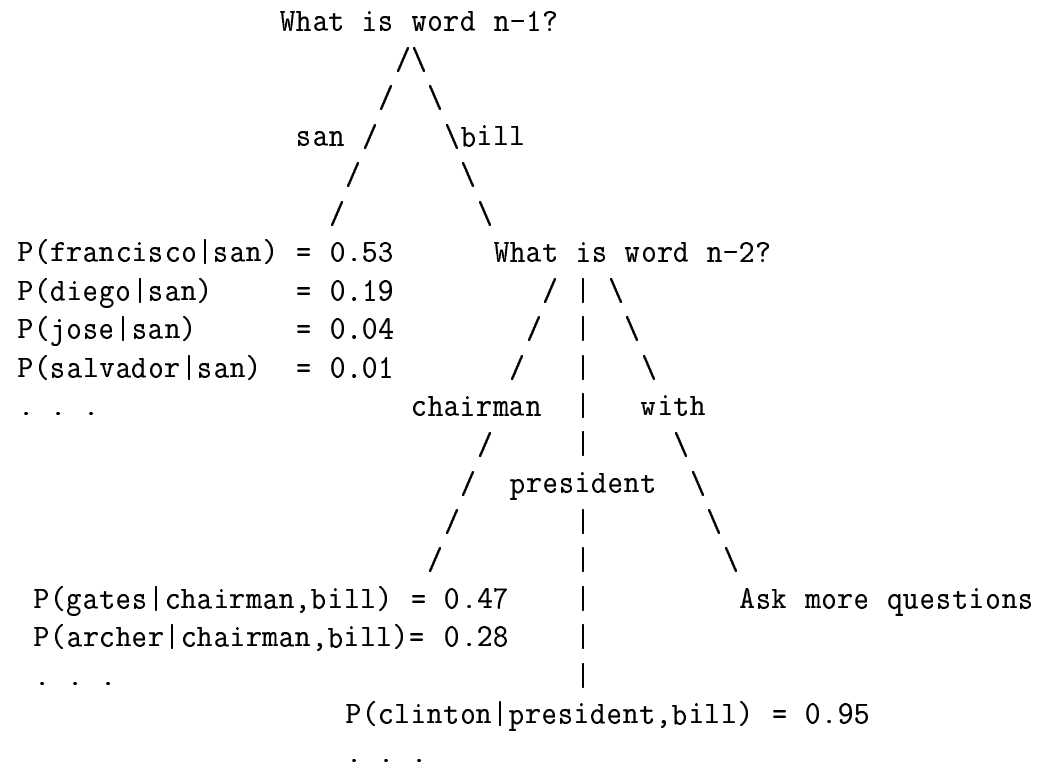

Figure 2.1: Decision Tree Example

- 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:

$$\mathcal{W} \equiv \{\text{All possible answers to question } W\} \quad (2.1)$$

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

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.