Machine Learning 34, 211–231 (1999)

An Algorithm that Learns What's in a Name

DANIEL M. BIKEL* dbikel@seas.upenn.edu
RICHARD SCHWARTZ schwartz@bbn.com
RALPH M. WEISCHEDEL weisched@bbn.com
BBN Systems and Technologies, 70 Fawcett Street, Cambridge MA 02138

Editors: Claire Cardie and Raymond Mooney

Abstract. In this paper, we present IdentiFinder™, a hidden Markov model that learns to recognize and classify names, dates, times, and numerical quantities. We have evaluated the model in English (based on data from the Sixth and Seventh Message Understanding Conferences [MUC-6, MUC-7] and broadcast news) and in Spanish (based on data distributed through the First Multilingual Entity Task [MET-1]), and on speech input (based on broadcast news). We report results here on standard materials only to quantify performance on data available to the community, namely, MUC-6 and MET-1. Results have been consistently better than reported by any other learning algorithm. IdentiFinder's performance is competitive with approaches based on handcrafted rules on mixed case text and superior on text where case information is not available. We also present a controlled experiment showing the effect of training set size on performance, demonstrating that as little as 100,000 words of training data is adequate to get performance around 90% on newswire. Although we present our understanding of why this algorithm performs so well on this class of problems, we believe that significant improvement in performance may still be possible.

Keywords: named entity extraction, hidden Markov models

1. The named entity problem and evaluation

1.1. The named entity task

The named entity task is to identify all named locations, named persons, named organizations, dates, times, monetary amounts, and percentages in text (see figure 1). Though this sounds clear, enough special cases arise to require lengthy guidelines, e.g., when is *The Wall Street Journal* an artifact, and when is it an organization? When is *White House* an organization, and when a location? Are branch offices of a bank an organization? Is a street name a location? Should *yesterday* and *last Tuesday* be labeled dates? Is *mid-morning* a time? In order to achieve human annotator consistency, guidelines with numerous special cases have been defined for the Seventh Message Understanding Conference, MUC-7 (Chinchor, 1998).

Both the boundaries of an expression and its label must be marked. The Standard Generalized Markup Language, or SGML, is an abstract syntax for marking information and

Current address: Department of Computer and Information Science, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104.

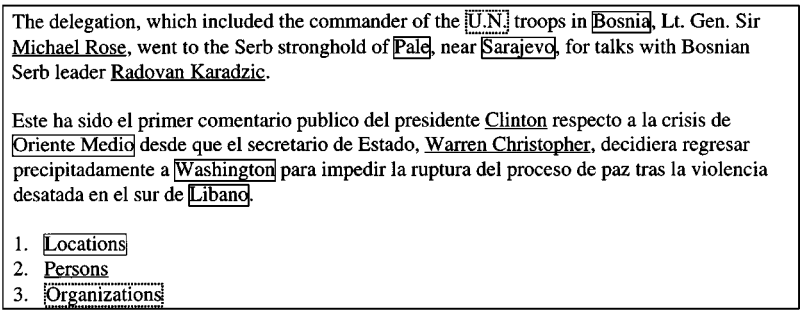

Figure 1. Examples (examples of correct labels for English text and for Spanish text).

structure in text, and is therefore appropriate for named entity mark-up. Various GUIs to support manual preparation of answer keys are available.

1.2. Evaluation metric

A computer program is used to evaluate the performance of a name-finder, called a "scoring program". The scoring program developed for the MUC and Multilingual Entity Task (MET) evaluations measures both precision (P) and recall (R), terms borrowed from the information-retrieval community, where

$$P = \frac{\text{number of correct responses}}{\text{number of responses}} \quad \text{and} \quad R = \frac{\text{number of correct responses}}{\text{number correct in key}}. \tag{1.1}$$

(The term *response* is used to denote "answer delivered by a name-finder"; the term *key* or *key file* is used to denote "an annotated file containing correct answers".) Put informally, recall measures the number of "hits" vs. the number of possible correct answers as specified in the key, whereas precision measures how many answers were correct ones compared to the number of answers delivered. These two measures of performance combine to form one measure of performance, the F-measure, which is computed by the uniformly weighted harmonic mean of precision and recall:

$$F = \frac{RP}{\frac{1}{2}(R+P)}. \tag{1.2}$$

In MUC and MET, a correct answer from a name-finder is one where the label and both boundaries are correct. There are three types of labels, each of which use an attribute to specify a particular entity. Label types and the entities they denote are defined as follows:

1. entity (ENAMEX): person, organization, location
2. time expression (TIMEX): date, time
3. numeric expression (NUMEX): money, percent.

A response is half-correct if the label (both type and attribute) is correct but only one boundary is correct. Alternatively, a response is half-correct if only the type of the label (and not the attribute) and both boundaries are correct. Automatic scoring software is available, as detailed in Chinchor (1998).

2. Why

2.1. Why the named entity (NE) problem

First and foremost, we chose to work on the named entity (NE) problem because it seemed both to be solvable and to have applications. The NE problem has generated much interest, as evidenced by its inclusion as an understanding task to be evaluated in both the Sixth and Seventh Message Understanding Conferences (MUC-6 and MUC-7) and in the First and Second Multilingual Entity Task evaluations (MET-1 and MET-2). Furthermore, at least one commercial product has emerged: NameTag™ from IsoQuest. The NE task had been defined by a set of annotator guidelines, an evaluation metric and example data (Sundheim & Chinchor, 1995).

Second, though the problem is relatively easy in mixed case English prose, it is a challenge in cases where case does not signal proper nouns, e.g., in Chinese, Japanese, German or non-text modalities (e.g., speech). Since the task was generalized to other languages in the Multilingual Entity Task (MET), the task definition is no longer dependent on the use of mixed case in English. Figure 2 shows some difficulties involved in name recognition in unicase English, using corporation names for illustration. All of the examples are taken from on-line newswire text studied. The first example is the easiest; a key word (*CO.*) strongly indicates the existence of a company name. However, the full, proper form will not always be used; example 2 shows a short form, an alias. Many shortened forms are algorithmically predictable. Example 3 illustrates a third easy case, the introduction of an acronym. Examples 1–3 are all handled well in the state of the art. Examples 4–6 are far more challenging, and call for improved performance. For instance, in examples 4 and 5 there is no clue in the names that they are company names; the underlined context in which they occur is the critical clue to recognizing that a name is present. In example 6, the

1. **MATSUSHITA ELECTRIC INDUSTRIAL CO.** HAS REACHED AGREEMENT …
2. IF ALL GOES WELL, **MATSUSHITA** AND ROBERT BOSCH WILL …
3. **VICTOR CO. OF JAPAN (JVC)** AND SONY CORP. …
4. IN A FACTORY OF **BLAUPUNKT WERKE**, A **ROBERT BOSCH** SUBSIDIARY, …
5. **TOUCH PANEL SYSTEMS**, CAPITALIZED AT 50 MILLION YEN, IS OWNED …
6. **MATSUSHITA EILL** DECIDE ON THE PRODUCTION SCALE. …

Figure 2. English examples (finding names ranges from the easy to the challenging. Company names are in boldface. The underlined text is discussed below).

problem is an error in the text itself; the challenge is recognizing that *MATSUSHITA EILL* is not a company, but that *MATSUSHITA* is.

A third motivation for our working on the NE problem is that it is representative of a general challenge for learning: given a set of concepts to be recognized and labeled, how can an algorithm learn to reliably spot and label new examples of the concepts? Although we have primarily applied our approach to the NE problem, we have begun to tackle additional term classification tasks. All classes are contiguous sequences of words where local context usually contains enough information to identify the term as a class member.

2.2. Why a learning algorithm

Most current techniques for named entity recognition are based on handcrafting finite state patterns to recognize names, dates, etc. (Appelt et al., 1995; Weischedel, 1995). Unfortunately, straightforward rules, such as

```
<proper-noun>+ <corporate designator> ==> <corporation>
```

are not nearly adequate for state-of-the-art performance, nor do they capture typical naming conventions. For instance, humans would have no problem predicting not only that "BOSTON POWER & LIGHT" is a corporation name but also that it is an electric utility, even though they have never heard that name before. (There is no such company.) In fact, organizations tend to choose names that identify the type of business/government purpose they have. The chance to eliminate requiring your best people to pour over data to find rules to achieve state-of-the-art performance is strong motivation to begin research in learning algorithms.

Our previous experience with handwritten rules is that each new source of text requires significant tweaking of rules to maintain optimal performance. That is, tackling the newspaper sources of the New York Times newswire after developing a rule set for the Wall Street Journal requires significant hand tuning. Even if the technology is good enough to be embedded in various applications, the maintenance costs for handcrafted rule systems could be quite steep. Furthermore, moving to other modalities such as speech input, or merely to upper case text, may require substantial modification of rule sets to obtain optimal performance for each modality.

In our earlier experience with handcrafted rules, we found that rules for one language may help very little in developing rule sets for another language. While the English rule set was suggestive for developing a rule set for Spanish, virtually nothing carried over to a rule set for Chinese.

3. A hidden Markov model

Name recognition may be viewed as a classification problem, where every word is either part of some name (such as the seven types of named entities in the MUC evaluations described earlier) or not part of any name. In recent years, hidden Markov models (HMMs) have enjoyed great success in other textual classification problems—most notably part-of-speech tagging (Church, 1988; Weischedel et al., 1993). Given this success, and given the locality of phenomena which indicate names in text, such as titles like "Mr." preceding a

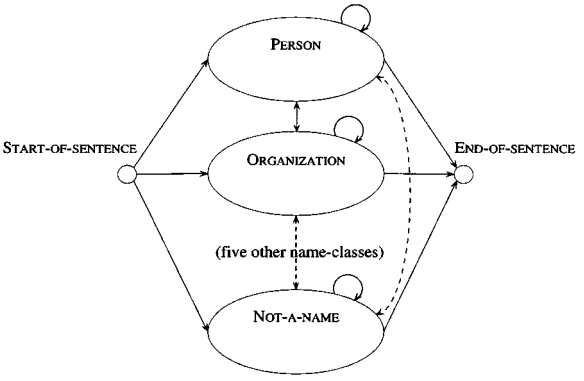

Figure 3. Pictorial representation of conceptual model (the subgraph of name-classes is complete, indicated here by the dashed arcs).

person name, we chose to develop a variant of an HMM for the name recognition task. (See Rabiner (1989) for an excellent tutorial on HMMs.)

By definition of the task, only a single label can be assigned to a word in context. Therefore, our model will assign to every word either one of the desired classes or the label NOT-A-NAME to represent "none of the desired classes". We organize the states of this HMM-variant into regions, one region for each desired class plus one for NOT-A-NAME. See figure 3. The HMM will have a model of each desired class and of the other text. The implementation is not confined to the seven classes of NE; an arbitrary number of classes may be provided to the system at run-time. Additionally, there are two special states, the START-OF-SENTENCE and END-OF-SENTENCE states.

Within each of the regions, we use a model for computing the likelihood of words occurring within that region (name-class), called a statistical bigram language model. A statistical bigram language model computes the likelihood of a sequence of words by employing a Markov chain, where every word's likelihood is based simply on the previous word. More formally, every word is represented by a state in the bigram model, and there is a probability associated with every transition from the current word to the next word. To determine the likelihood of a sequence of words w_1 through w_n, the model computes $\prod_{i=1}^{n} p(w_i \mid w_{i-1})$ (a special +begin+ word is used to compute the likelihood of w_1). The use of a statistical bigram model in each name-class means that the number of states in each of the name-class regions is equal to the vocabulary size, $|V|$.

For the purposes of name-finding, we must find the most likely sequence of name-classes ($\boldsymbol{NC}$) given a sequence of words ($\boldsymbol{W}$):

$$\max \Pr(\boldsymbol{NC} \mid \boldsymbol{W}) \tag{3.1}$$

We assume a generative model, i.e., that the HMM generates the sequence of words and labels. We use Bayes' Rule:

$$\Pr(\boldsymbol{NC} \mid \boldsymbol{W}) = \frac{\Pr(\boldsymbol{W}, \boldsymbol{NC})}{\Pr(\boldsymbol{W})} \tag{3.2}$$

Since the unconditioned probability of the word sequence—the denominator—is constant for any given sentence, we can maximize the right-hand side of Eq. (3.2) by maximizing the numerator alone. We will describe how Pr(***W***, ***NC***) is modeled in Section 3.2.1.

The numerator of Eq. (3.2) is the joint probability of the word and name-class sequence. As is necessary with a Markov model, we make independence assumptions when computing this joint probability. Accordingly, the generation of words and name-classes proceeds in three steps:

1. Select a name-class *NC*, conditioning on the previous name-class and the previous word.
2. Generate the first word inside that name-class, conditioning on the current and previous name-classes.
3. Generate all subsequent words inside the current name-class, where each subsequent word is conditioned on its immediate predecessor (as per a standard bigram language model).

These three steps are repeated until the entire observed word sequence is generated. Using the Viterbi algorithm (Viterbi, 1967), we efficiently search the entire space of all possible name-class assignments, maximizing the numerator of Eq. (3.2), Pr(***W***, ***NC***).

Let us illustrate the high-level computation of the likelihood of a word-name-class sequence with an example. Suppose IdentiFinder encounters the sentence

```
Mr. Jones eats.
```

According to rules of MUC and MET, the correct annotation for such a sentence is

```
Mr. <ENAMEX TYPE=PERSON>Jones</ENAMEX> eats.
```

That is, the token `Jones` is in the PERSON name-class, while the other tokens are in the NOT-A-NAME name-class. The model would assign the following likelihood to this word-name-class sequence (which we would hope to be the most likely, given sufficient training):

Pr(NOT-A-NAME | START-OF-SENTENCE, "+end+")*
Pr("Mr." | NOT-A-NAME, START-OF-SENTENCE) *
Pr(+end+ | "Mr.", NOT-A-NAME) *
Pr(PERSON | NOT-A-NAME, "Mr.") *
Pr("Jones" | PERSON, NOT-A-NAME) *
Pr(+end+ | "Jones", PERSON) *
Pr(NOT-A-NAME | PERSON, "Jones") *
Pr("eats" | NOT-A-NAME, PERSON) *
Pr("." | "eats", NOT-A-NAME) *
Pr(+end+ | ".", NOT-A-NAME) *
Pr(END-OF-SENTENCE | NOT-A-NAME, ".")

The details of these probability estimates as well as the use of the special +end+ word are described below, in Section 3.2.

Informally, the construction of the model in this manner indicates that we view each type of "name" to be its own language, with separate bigram probabilities for generating its words. This reflects our intuition of the following:

- There is generally predictive internal evidence regarding the class of a desired entity. Consider the nature of stereotypical names for airlines, utilities, other corporations and government organizations. In many cultures, first person names are stereotypical; in Chinese, family names are stereotypical. In Chinese and Japanese, special characters are used to transliterate foreign names.
- Local external evidence often suggests the boundaries and class of one of the desired expressions. Titles signal beginnings of person names. Closed class words, such as determiners, pronouns, and prepositions often signal a boundary.

While the number of word-states within each name-class is equal to $|V|$, this "interior" bigram language model is ergodic, i.e., there is a probability associated with every one of the $|V|^2$ transitions. As a parameterized, trained model, if such a transition were never observed, the model "backs off" to a less powerful model, as described below, in Section 3.2.3.

3.1. Words and word-features

The word feature is the one part of this model that is language-dependent. Fortunately, the word feature computation is an extremely small part of the implementation, at roughly twenty lines of code. The rationale for having such features is clear:

- In Roman languages, capitalization gives good evidence of names.[1]
- Numeric symbols can automatically be grouped into categories, as in the initial features in Table 1.
- Semantic classes can be defined by lists of words having a semantic feature.
- Special character sets such as the ones used for transliterating names in Chinese or in Japanese can be identified.

Throughout most of the model, we consider words to be ordered pairs (or two-element vectors), composed of word and word-feature, denoted $\langle w, f \rangle$. The word feature is a simple, deterministic computation performed on each word as it is added to or looked up in the vocabulary. It produces one of the fourteen values in Table 1.

These values are computed in the order listed, so that in the case of non-disjoint feature-classes, such as `containsDigitAndAlpha` and `containsDigitAndDash`, the former will take precedence. The first eight features arise from the need to distinguish and annotate monetary amounts, percentages, times and dates. The rest of the features distinguish types of capitalization and all other words (such as punctuation marks, which are separate tokens). In particular, the `firstWord` feature arises from the fact that if a word is capitalized and is the first word of the sentence, we have no good information as to why it is capitalized (but note that `allCaps` and `capPeriod` are computed before `firstWord`, and therefore take precedence).

Table 1. Word features, examples and intuition behind them.[2]

Word feature	Example text	Intuition
`twoDigitNum`	90	Two-digit year
`fourDigitNum`	1990	Four-digit year
`containsDigitAndAlpha`	A8956-67	Product code
`containsDigitAndDash`	09-96	Date
`containsDigitAndSlash`	11/9/89	Date
`containsDigitAndComma`	23,000.00	Monetary amount
`containsDigitAndPeriod`	1.00	Monetary amount, percentage
`otherNum`	456789	Other number
`allCaps`	BBN	Organization
`capPeriod`	M.	Person name initial
`firstWord`	*first word of sentence*	No useful capitalization information
`initCap`	Sally	Capitalized word
`lowerCase`	can	Uncapitalized word
`other`	,	Punctuation marks, all other words

In the early stages of IdentiFinder's development, the word-feature was not in the model; instead the system relied on a third-level back-off part-of-speech tag, which in turn was computed by our stochastic part-of-speech tagger. The tags were taken at face value: there were no k-best tags; the system treated the part-of-speech tagger as a "black box". Although the part-of-speech tagger used capitalization to help it determine proper-noun tags, this feature was only implicit in the model, and then only after two levels of back-off. Also, the capitalization of a word was submerged in the muddiness of part-of-speech tags, which can "smear" the capitalization probability mass over several tags. Because it seemed that capitalization would be a good name-predicting feature, and that it should appear earlier in the model, we eliminated the reliance on part-of-speech altogether, and opted for the more direct, word-feature model described above, in Section 3. Also in the early stages of development, we had a very small number of features, indicating whether the word was a number, the first word of a sentence, all uppercase, initial-capitalized or lower-case. We then expanded the feature set to its current state in order to capture more subtleties related mostly to numbers; due to increased performance (although not entirely dramatic) on every test, we kept the enlarged feature set.

3.2. Formal model

This section describes the model formally, discussing the transition probabilities to the word-states, which "generate" the words of each name-class. As with most trained, probabilistic models, we have a most accurate, most powerful model, which will "back off" to a less-powerful model when there is insufficient training, and ultimately back-off to unigram probabilities.

3.2.1. Top level model. The top-level model consists of three components: (1) a model to generate a name-class, (2) a model to generate the first word in a name-class and (3) a model to generate all subsequent words in a name-class.

In order to generate the first word, we must make a transition from one name-class to another, as well as calculate the likelihood of that word. Our intuition was that a word preceding the start of a name-class (such as "Mr.", "President" or other titles preceding the PERSON name-class) and the word following a name-class would be strong indicators of the subsequent and preceding name-classes, respectively. Accordingly, the probability for generating the first word of a name-class is factored into two parts:

$$\Pr(NC \mid NC_{-1}, w_{-1}) \cdot \Pr(\langle w,\ f\rangle_{\text{first}} \mid NC, NC_{-1}). \tag{3.3}$$

The top level model for generating all but the first word in a name-class is

$$\Pr(\langle w,\ f\rangle \mid \langle w,\ f\rangle_{-1}, NC). \tag{3.4}$$

There is also a distinguished "+end+" word, so that the probability may be computed for any current word to be the final word of its name-class, i.e.,

$$\Pr(\langle +end+, \texttt{other}\rangle \mid \langle w,\ f\rangle_{\text{final}}, NC). \tag{3.5}$$

As one might imagine, it would be useless to have the first factor in Eq. (3.3) be conditioned on the +end+ word, so the probability is conditioned on the previous *real* word of the previous name-class, i.e., we compute

$$\Pr(NC \mid NC_{-1}, w_{-1}) \begin{cases} w_{-1} = +end+ \text{ if} \\ \quad NC_{-1} = \text{START-OF-SENTENCE} \\ w_{-1} = \text{last observed word otherwise} \end{cases} \tag{3.6}$$

Note that the above probability is not conditioned on the word-feature of w_{-1}. Our intuition is that in cases where the previous word would help the model predict the next name-class, the word feature—capitalization in particular—is not important: "Mr." is a good indicator of the next word beginning the PERSON name-class, regardless of capitalization, especially since it is almost never seen as "mr.".

3.2.2. Training: Estimating probabilities. The calculation of the above probabilities is straightforward, using events/sample-size:

$$\Pr(NC \mid NC_{-1}, w_{-1}) = \frac{c(NC, NC_{-1}, w_{-1})}{c(NC_{-1}, w_{-1})} \tag{3.7}$$

$$\Pr(\langle w,\ f\rangle_{\text{first}} \mid NC, NC_{-1}) = \frac{c(\langle w,\ f\rangle_{\text{first}}, NC, NC_{-1})}{c(NC, NC_{-1})} \tag{3.8}$$

$$\Pr(\langle w,\ f\rangle \mid \langle w,\ f\rangle_{-1}, NC) = \frac{c(\langle w,\ f\rangle, \langle w,\ f\rangle_{-1}, NC)}{c(\langle w,\ f\rangle_{-1}, NC)} \tag{3.9}$$

where $c()$ represents the number of times the events occurred in the training data (the *count*).

3.2.3. Back-off models and smoothing. Ideally, we would have sufficient training (or at least one observation of) every event whose conditional probability we wish to calculate. Also, ideally, we would have sufficient samples to estimate the conditional probabilities, e.g., for $\Pr(NC \mid NC_{-1}, w_{-1})$, we would like to have seen sufficient numbers of NC_{-1}, w_{-1}. Unfortunately, there is rarely enough training data to estimate accurate probabilities for all possibilities. There are two cases: words not seen in training ("unknown words") and other events where insufficient training data is available.

3.2.3.1. Unknown words. The vocabulary of the system is built as it trains. Necessarily, then, the system knows about all words for which it stores bigram counts in order to compute the probabilities in Eqs. (3.3)–(3.5). All unknown words are mapped to the token _UNK_. The question arises how the system should estimate probabilities involving _UNK_, since there are three ways in which they can appear in a bigram: as the current word, as the previous word or as both. A good answer is to train a separate, unknown word-model on held-out data, to gather statistics of unknown words occurring in the midst of known words. That is to say, since we treat all unknown words as one token _UNK_, we would like to estimate how _UNK_ occurs in some of our training data.

Ideally, one would hold out one article at a time for smoothing or unknown word-training. However, to simplify the implementation, we hold out 50% of our data to train the unknown word model (the vocabulary is built up on the first 50%), save these counts in a training data file, then hold out the other 50% and concatenate these bigram counts with the first unknown word training file. This way, we can gather likelihoods of an unknown word appearing in a bigram using all available training data. This approach is perfectly valid, as we are trying to estimate that which we have not legitimately seen in training. When performing name recognition, if either word of the bigram is unknown, the model used to estimate the probabilities of Eqs. (3.3)–(3.5) is the unknown word model, otherwise it is the model from the normal training. The unknown word-model can be viewed as a first level of back-off, therefore, since it is used when an unknown word is encountered, and is necessarily not as accurate as the bigram model formed from the actual training.

3.2.3.2. Further back-off models and smoothing. Whether a bigram contains an unknown word or not, it is possible that either model may not have seen this bigram. Table 2 shows a graphic illustration of the back-off scheme.

The weight for each back-off model is computed on-the-fly, using the following formula: if computing $\Pr(X \mid Y)$ for some events X and Y, assign weight of λ to the direct estimate of this conditional probability computation (using one of the formulae of Section 3.2.2) and a weight of $(1 - \lambda)$ to the back-off model, where

$$\lambda = \left(1 - \frac{\text{old } c(Y)}{c(Y)}\right) \frac{1}{1 + \frac{\text{unique outcomes of } Y}{c(Y)}}. \tag{3.10}$$

Table 2. Back-off strategy.

Name-class bigrams	First-word bigrams	Non-first-word bigrams
$\Pr(NC \mid NC_{-1}, w_{-1})$	$\Pr(\langle w,\ f\rangle_{\text{first}} \mid NC, NC_{-1})$	$\Pr(\langle w,\ f\rangle \mid \langle w,\ f\rangle_{-1}, NC)$
$\vdots$	$\vdots$	$\vdots$
$\Pr(NC \mid NC_{-1})$	$\Pr(\langle w,\ f\rangle \mid \langle$ *+begin+*,`other`$\rangle, NC)$	$\Pr(\langle w,\ f\rangle \mid NC)$
$\vdots$	$\vdots$	$\vdots$
$\Pr(NC)$	$\Pr(\langle w,\ f\rangle \mid NC)$	$\Pr(w \mid NC) \cdot \Pr(f \mid NC)$
$\vdots$	$\vdots$	$\vdots$
$\frac{1}{\text{number of name-classes}}$	$\Pr(w \mid NC) \cdot \Pr(f \mid NC)$	$\frac{1}{\lvert V\rvert} \cdot \frac{1}{\text{number of word features}}$
	$\vdots$	
	$\frac{1}{\lvert V\rvert} \cdot \frac{1}{\text{number of word features}}$	

The quantity "old $c(Y)$" is the sample size of the model from which we are backing off. This is a rather simple method of smoothing, which tends to work well when there are only three or four levels of back-off.[3] This method also overcomes the problem when a back-off model has roughly the same amount of training as the current model, via the first factor of Eq. (3.10), which essentially ignores the back-off model and puts all the weight on the primary model, in such an equi-trained situation. For example, suppose we are trying to compute the back-off weight λ for a computation of Pr(PERSON | NOT-A-NAME, "Mr."), but it turns out that c(NOT-A-NAME, "Mr.") $\approx c$(NOT-A-NAME). In such a case, we would like to effectively remove the estimate of Pr(PERSON | NOT-A-NAME) from the smoothing of our estimate of Pr(PERSON | NOT-A-NAME, "Mr."), since the two have equal training, but the latter is a superior model, since it conditions on more context. This is exactly what happens with the first factor in Eq. (3.10), since, when we are computing Pr(PERSON | NOT-A-NAME), old $c(Y) \approx c(Y)$, making λ almost zero and pushing all the rest of the probability mass to the computation of Pr(PERSON).

The second factor of Eq. (3.10), which does the real work of smoothing, is based on the notion that the number of unique outcomes over the sample size is a crude measure of the certainty of the model. This "certainty" is essentially a measure (in the (0, 1) interval) of the uniformity of the distribution from state Y, where a completely uniform distribution has the most uncertainty.

As an example—disregarding the first factor—if we saw the bigram "come hither" once in training and we saw "come here" three times, and nowhere else did we see the word "come" in the NOT-A-NAME class, when computing

Pr("hither" | "come", NOT-A-NAME),

we would back off to the unigram probability

Pr("hither" | NOT-A-NAME)

with a weight of $\frac{1}{3}$, since the number of unique outcomes for the word-state for "come" would be two, and the total number of times "come" had been the preceding word in a bigram would be four (a weight of $1/(1+\frac{2}{4}) = \frac{2}{3}$ for the bigram probability, a weight of $1 - \frac{2}{3} = \frac{1}{3}$ for the back-off model).

3.3. Comparison with a traditional HMM

Unlike a traditional HMM, the probability of generating a particular word is 1 for each word-state inside each of the name-class states. An alternative—and more traditional—model would have a small number of states within each name-class, each having, perhaps, some semantic significance, e.g., three states in the PERSON name-class, representing a first, middle and last name, where each of these three states would have some probability associated with emitting any word from the vocabulary. We chose to use a bigram language model because, while less semantically appealing, such n-gram language models work remarkably well in practice. Also, as a first research attempt, an n-gram model captures the most general significance of the words in each name-class, without presupposing any specifics of the structure of names, á la the PERSON name-class example, above. More important, either approach is mathematically valid, as long as all transitions out of a given state sum to one.

3.4. Decoding: The recognition process

The number of possible state sequences for N states in an ergodic model for a sentence of m words is N^m. However, using dynamic programming and an appropriate merging of multiple theories when they converge on a particular state—the Viterbi decoding algorithm—a sentence can be "decoded" in time linear to the number of tokens in the sentence, $O(m)$. Since we are interested in recovering the name-class state sequence, we pursue N theories, one for each name-class, at every given step of the algorithm (for MUC-6, e.g., $N = 8$).

4. Implementation

IdentiFinder is implemented in C++. On a desktop PC running linux with a 200 MHz processor and 128 MB RAM, IdentiFinder trains on the 650 K word training corpus of Wall Street Journal text in six minutes. Named entity recognition runs at about 12 MB/h.

Up until now, our effort has been focused on recognition quality (F-measure). We believe that substantial speed up is possible, and have begun implementation to achieve that.

5. Results of evaluation

In this section we report the results of evaluating the current version of the learning software. We report the results for English and for Spanish mixed case, and for English in varying

Table 3. F-measure scores (IdentiFinder's performance as compared to the best reported scores for each category).

	Language	Best rules	IdentiFinder
Mixed case	English (WSJ)	96.4	94.9
Upper case	English (WSJ)	89	93.6
Speech form	English (WSJ)	74	90.7
Mixed case	Spanish	93	90

modalities. In addition, we present the results of experiments that determine the impact of the training set size on the algorithm's performance in both English and Spanish. For each language, we have a held-out development test set and a held-out, blind test set. We only report results on the blind test set for each respective language.

5.1. English and Spanish results

Our test set for English data is that from MUC-6, a collection of 30 *Wall Street Journal* documents (we used a different test set during development).[4] Our Spanish test set is from materials used in MET-1, comprised of articles from the news agency AFP.[5]

Table 3 compares the F-measure for IdentiFinder against the highest performing system of handcrafted rules.[6] We can make the following two observations.

1. Performance of the best hand-crafted system on mixed case is better than that of the HMM. However, the performance is close enough that we choose the learning approach rather than requiring computational linguists to maintain rule sets. Indeed, significance tests (see Chinchor, 1995) indicate that IdentiFinder performs comparably to the best hand-crafted system on mixed-case text.
2. Spanish performance trails that of English. There are several contributing factors to this performance discrepancy:

   - Three times more training data were available for English. Spanish performance should improve with more training data.
   - The training data for Spanish seems to have more internal inconsistencies. Another pass through the data to minimize inconsistencies should improve performance.
   - Many words in the Spanish names are lower case, thereby making name-finding in Spanish somewhat harder than in English. For example, *departamento* ("Department") could often start an organization name, and nationality adjectives, such as *coreana* ("Korean") could appear in names and by convention are not capitalized.
   - The domain of the Spanish data, press conferences, is probably somewhat harder than the narrower domain of English, namely the change of corporate officers in major corporations.

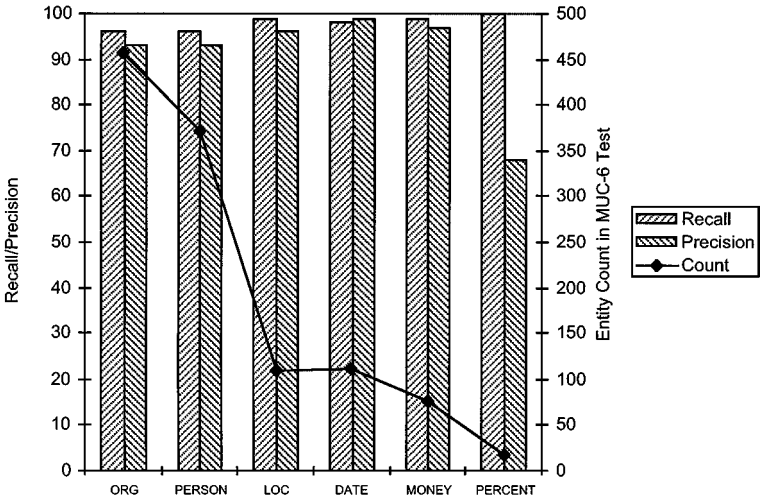

Figure 4. Detailed performance analysis of IdentiFinder on the MUC-6 test. The left axis is percentage of recall/precision for the bars; the right axis is the raw entity count for the line graph. There are no occurrences of the TIME name-class in the MUC-6 test.

5.2. Detailed performance results on English

While the F-measure is convenient as a simple, single metric of name-finding performance, it is illuminating and indeed necessary when evaluating a name-finder to look at the precision and recall from which the F-measure is derived, as well as the precision and recall for specific named entities. The overall recall for IdentiFinder on the MUC-6 test is 96%, the overall precision, 93%, which combine to form an F-measure of 94.92.

Figure 4 illustrates the performance of IdentiFinder on the various named entities that appeared in the MUC-6 test, giving separate recall and precision scores for each type of entity. It is important to note that not all entity types appear with the same frequency, and therefore performance on the far more numerous entity types has a much greater impact on overall performance than on the infrequent ones. The combined line chart serves to illustrate the relative frequency of the entities, using the absolute scale on the right-hand side of the chart; e.g., there were 457 organization (ORG) names in the MUC-6 test, but only 76 money amounts. All data for figure 4 were generated by the MUC-7 scoring program running in MUC-6 compatibility mode.

5.3. Effect of word features

As the word-feature computation is the only part of the model that is language-dependent, it is instructive to evaluate the performance of IdentiFinder with only the smallest subset of word-features vs. the entire set. Early in the system's development there were only five word features, to detect whether a word was a number, all upper-case, all lower-case, the first word of a sentence or none of the above. Using only these five features, IdentiFinder

scored an F of 94.07 (96% recall, 92% precision) on the MUC-6 test, as compared to the 94.92 F-measure score with the full feature set. This experiment indicated that while the full word feature computation yields better performance, it is not nearly as significant as the other parts of the model.

5.4. Modalities other than mixed case prose

For English, if mixed case is not available, the NE task becomes significantly harder, even for humans. In figure 5, we show the same sentence in mixed case, in upper case, and in the format generated by automatic speech recognition, termed "SNOR" format. SNOR format not only lacks case, but lacks all punctuation, and has all numbers spelled out as words.

Figure 6 illustrates how one mixed case training set can serve to train IdentiFinder for uppercase input, for speech (SNOR) format, and for OCR input. It is easy to see that one can automatically convert the mixed case training set into upper case training, and then retrain IdentiFinder on the uppercase material. However, it is also easy to convert mixed case prose into SNOR format by upcasing the text, removing all punctuation, and converting

Mixed Case:	The British company, whose interests include the Cunard cruise lines and the Ritz hotel of London, said Simon Keswick, the head of Hong Kong Land Holdings Ltd. will take over as chairman May 26. The current acting chairman, Alan, will become a deputy chairman of the group.
Upper Case:	THE BRITISH COMPANY, WHOSE INTERESTS INCLUDE THE CUNARD CRUISE LINES AND THE RITZ HOTEL OF LONDON, SAID SIMON KESWICK, THE HEAD OF HONGKONG LAND HOLDINGS LTD. WILL TAKE OVER AS CHAIRMAN MAY 26. THE CURRENT ACTING CHAIRMAN, ALAN CLEMENTS, WILL BECOME A DEPUTY CHAIRMAN OF THE GROUP.
SNOR:	THE BRITISH COMPANY WHOSE INTERESTS INCLUDE THE CUNARD CRUISE LINES AND THE RITZ HOTEL OF LONDON SAID SIMON KESWICK THE HEAD OF HONGKONG LAND HOLDINGS LIMITED WILL TAKE OVER AS CHAIRMAN MAY TWENTY SIX THE CURRENT ACTING CHAIRMAN ALAN CLEMENTS WILL BECOME A DEPUTY CHAIRMAN OF THE GROUP

Figure 5. Three modalities (the task becomes increasingly difficult as one moves from mixed case to upper case, to SNOR format).

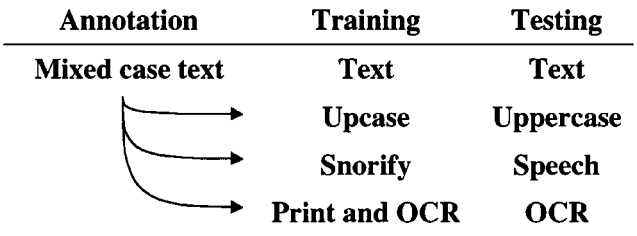

Figure 6. Deriving alternative modalities (given mixed case training data, one can automatically derive training data for upper case text, for speech (SNOR) format and for optical character recognition (OCR)).

all numbers to their word form, e.g., 150 becomes *one hundred fifty*. For OCR, print the training files, then OCR it, and retrain.

We have conducted experiments on upper case text and SNOR format, and have planned to run experiments with OCR data in 1999. Results for the MUC-6 test materials are reported in Table 3. Two points should be noted about the results with respect to alternative modalities. First, the best score previously reported for an upper case version of MUC-6 was by the handcrafted rule system NameTag™, the same system that scored best on the mixed case version (Krupka, 1995). It had been designed to perform well on mixed case and on upper case. Second, the only performance by a rule-based system ever reported on SNOR format was by our own rule-based system. It should be noted that we had not prepared it specifically for SNOR, but had tuned it to handle upper case text. We estimate that a month's effort would have improved its scores by about 10 points of F. Given the performance of IdentiFinder without any labor for SNOR input, we had no motivation to spend a person month to get an exact score for the rule-based system. Nevertheless, the following two conclusions were very apparent.

1. IdentiFinder clearly outperformed all previous approaches when mixed case was not available.
2. IdentiFinder required no labor to handle upper case text or speech format. It required only a few machine cycles to convert the mixed case training data to other forms and retrain.

5.5. The amount of training data required

With any learning technique one of the important questions is how much training data is required to get acceptable performance. More generally, how does performance vary as the training set size is increased or decreased? We ran a sequence of experiments in English and in Spanish to try to answer this question for the model that was implemented.

For English, there were 650,000 words of training data. By that we mean that the text of the document itself (including headlines but not including SGML tags) was 650,000 words long. Given this maximum size of training available to us, we successively divided the training material in half until we were using only 60,000 words for the smallest experiment.

The results are shown in figure 7 below using a logarithmic scale for the x-axis. The positive outcome of the experiment is that half as much training data would have given almost equivalent performance. Had we used only one sixth of the data or approximately 100,000 words, performance would have degraded slightly, only about 1–2 percent. Reducing the training set to 60,000 words would have had a more significant decrease in the performance of the system; however, the performance is still very high even with such a small training set.

With increased training data it would be possible to use even more detailed models that require more data and could achieve significantly improved overall system performance with those more detailed models. For Spanish we had only 223,000 words of training data. We also measured the performance of the system with half the training data or slightly more than 100,000 words of text. Figure 7 shows the results. There is almost no change in performance by using as little as 100,000 words of training data. Therefore, the results

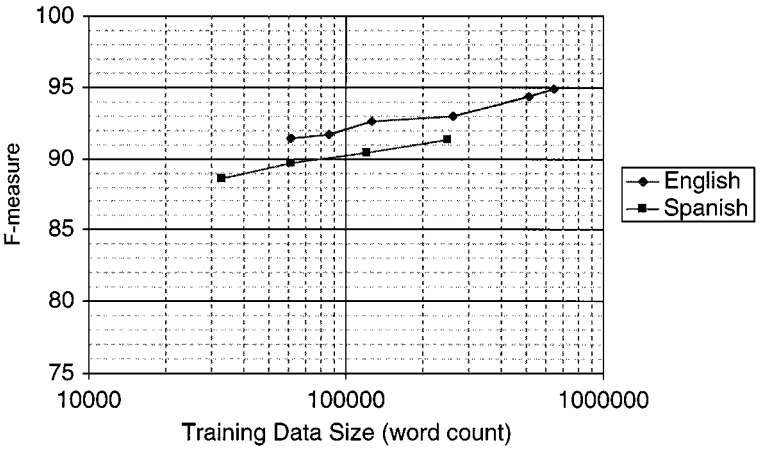

Figure 7. Impact of various training set sizes on performance (the learning algorithm performs remarkable well in both English and Spanish, nearly comparable to handcrafted systems with as little as 100,000 words of training data).

in both languages were comparable. As little as 100,000 words of training data produces performance nearly comparable to handcrafted systems.

The cost of using any algorithm is always an issue. To give a sense of the size of 650,000 words, that is roughly two-thirds the length of one edition of the Wall Street Journal. Annotators—those who mark answer keys—can reliably mark WSJ-style text with the seven MUC categories at a rate of 3000 words per hour for inexperienced annotators, and roughly 5000 words per hour for experienced annotators, according to experiments performed at BBN. Using IdentiFinder itself, BBN has also explored several automatic means of correction/adjudication, and several bootstrapping methods that make the task of creating a sufficient amount of training data quite manageable. One of these methods is to have IdentiFinder test on its own training, catching an important albeit limited class of errors due to inconsistency (either intra- or inter-annotator). In our experience, inter-annotator consistency is roughly 95% or above for our typical text sources.

6. Further work and error analysis

6.1. Further work

While our initial results have been quite favorable, there is still much that can be done potentially to improve performance and completely close the gap between learned and rule- based name-finding systems. We would like to incorporate the following into the current model:

- a hierarchical model to capture nested names, e.g., *Bank of Boston*
- longer-distance information, to find names not captured by our bigram model
- training heuristics to supplement annotated data with large volumes of unmarked language.

```
     The Turkish company, <ENAMEX TYPE="LOCATION">Birgen
Air</ENAMEX>, was using the plane to fill a charter commitment
to a German company, ...
```

Figure 8. IdentiFinder output on a sentence from the MUC-7 dryrun data.

6.2. Error analysis

The second bullet in Section 6.1 is of particular interest, especially in the analysis of certain errors currently made by IdentiFinder. Let us look at the sentence that is shown in figure 8, part of the MUC-7 dryrun data.

The angle-bracketed items are SGML tags, which delimit and identify the type of names in the text. Note that `Birgen Air` has been labeled a LOCATION, when it is an ORGANIZATION (and is so marked in the key file). According to the tokenization rules of MUC-6, MET-1 and MUC-7, punctuation marks such as commas are treated as separate tokens, meaning that the "word" directly preceding `Birgen`—as far as IdentiFinder's model is concerned—is "`,`", and the same "word" directly follows "`Air`". This means that currently, IdentiFinder is incapable of using the slightly wider context of a trigram that would include the word "`company`" when predicting the beginning of this ORGANIZATION. As it happens, `Birgen` is an unknown word, and the capitalized word `Air` happens to have a very high unigram probability for appearing within a LOCATION, due to many training examples of "`Edwards Air Force Base`" and the like, which are considered LOCATIONs, not ORGANIZATIONs. (For the same reason, "`_UNK_ Air`" is a very likely bigram in a LOCATION in the unknown word model.) Therefore, with the current bigram model, IdentiFinder has determined that "`Birgen Air`" is a LOCATION, when it is actually an ORGANIZATION, an error that *could* be potentially prevented by using more context in the model. The downside of using more context, such as trigrams, is that in practice exponentially more training data is required.

As another approach, one might consider effectively "stripping" seemingly inconsequential tokens such as commas out of the data stream processed by IdentiFinder, re-inserting them after name-finding is complete. While such a technique might appear to be helpful, it is actually quite detrimental, as punctuation tokens—commas in particular—give great evidence of names, especially given the prevalence of LOCATIONs having the form "`City, State`". For example, in the text fragment "`Los Angeles, California`", the comma helps signal the end of the first LOCATION "`Los Angeles`" and the beginning of the next LOCATION "`California`". The compromise between a full trigram model and the status quo is to develop a special-case extension to the model. Indeed, very recently IdentiFinder's model has been augmented with this special case, so that *both* the punctuation token *and* its adjacent token can be used to determine the likelihood of a name.

7. Other learning approaches to name-finding

There have been relatively few attempts to apply learning algorithms to the task of named entity recognition. However, Brill's transformation-based learning algorithm (Brill, 1995) has been applied to the NE problem, as outlined in Aberdeen et al. (1995). Performance thus far reported lags that of IdentiFinder by about 10 F-measure points. Bennett, Aone,

and Lovell (1997) use binary decision trees for the NE task. The decision tree decides whether to insert a begin category mark, end category mark, or nothing at each point in the sequence of input words. Their F-measure scores thus far are about 91, contrasted with IdentiFinder's 94.9 on the same test material.

Like IdentiFinder, these two alternative learning approaches use statistics to make decisions. However, only IdentiFinder has a complete probabilistic model that

- governs all decisions and
- models the categories of interest and the residual input that is not of interest.

A very recent approach is the MENE system (Borthwick et al., 1998), which employs similar features to those of IdentiFinder to create a Maximum Entropy model for name-finding. Using MENE alone, the reported F-measure results on the MUC-7 dry-run and formal test sets range from the mid-80s to the early 90s. However, Borthwick et al. (1998) have very successfully combined MENE with other, rule-based name-finders to achieve results superior to the rule-based systems alone.

8. Conclusions

None of the formalisms or techniques presented in this paper is new; rather, applying an HMM to this task and the model itself are novel. We have shown that using a fairly simple probabilistic model, finding names and other numerical entities as specified by the MUC NE and MET tasks can be performed with "near-human performance", often likened to an F-measure of 95 or above. We have also shown that such a system can be trained efficiently. Furthermore, the system is largely language-independent: for example, although none of the authors speaks Spanish, we were able to develop a Spanish name-finder by training IdentiFinder on answer keys marked by native speakers. We also showed that the technique handles multiple modalities automatically when automatic methods exist to convert the training data for one modality into another. To our knowledge, our learned name-finding system has achieved a higher F-measure than any other learned NE system. Indeed, IdentiFinder performs as well as the top-performing systems, whether learned or handcrafted, on mixed case text and is superior to all previously reported results when case information is lacking.

Acknowledgments

The work reported here was supported in part by the Defense Advanced Research Projects Agency and the DoD Counterdrug Technology Development Program Office. Technical agents for part of this work were Fort Huachucha under contract number DABT63-94-C-0062 and Rome Laboratory under contract number F30602-95-C-0111. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the DoD Counterdrug Technology Development Program Office, or the United States Government.

Notes

1. Spanish has many lower-case words in organization names.
2. This table represents the feature set used at the time of MET-1. We have subsequently made additions, such as list membership features, that have helped marginally in certain domains. We have also allowed for multiple, non-disjoint feature vectors, which again provide only marginal improvement in overall performance.
3. Any more levels of back-off might require a more sophisticated smoothing technique, such as deleted interpolation. No matter what smoothing technique is used, one must remember that smoothing is the art of estimating the probability of that which is unknown (i.e., not seen in training).
4. Unfortunately, the MUC-6 test is not a good measure of domain-independent performance, since the MUC-6 materials were obtained by a keyword search intended to retrieve articles reporting a change in corporate officers.
5. The MET-1 data was selected by the Government retrieving articles about press conferences from AFP.
6. For Mixed Case and Upper Case, the "Best Rules" column results are those of NameTag^{TM} reported at MUC-6. The "Best Rules" result in the "Speech Form" test is from an earlier, rule-based name-finder developed at BBN (the rules were tuned for regular WSJ, not speech output). Finally, the "Best Rules" result in the Mixed Case Spanish was reported anonymously in (Merchant, Okurowski, & Chinchor, 1996).

References

Aberdeen, J., Burger, J., Day, D., Hirschman, L., Robinson, P., & Vilain, M. (1995). MITRE: Description of the Alembic system used for MUC-6. *Proceedings of the Sixth Message Understanding Conference (MUC-6)* (pp. 141–155). Columbia, Maryland: Morgan Kaufmann Publishers, Inc.

Appelt, D.E., Jerry, R.H., Bear, J., Israel, D., Kameyama, M., Kehler, A., Martin, D., Myers, K., & Tyson, M. (1995). SRI international FASTUS system MUC-6 test results and analysis. *Proceedings of the Sixth Message Understanding Conference (MUC-6)* (pp. 237–248). Columbia, Maryland: Morgan Kaufmann Publishers, Inc.

Bennett, S.W., Aone, C., & Lovell, C. (1997). Learning to tag multilingual texts through observation. *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing* (pp. 109–116). Providence, Rhode Island: Morgan Kaufmann Publishers, Inc.

Borthwick, A., Sterling, J., Agichtein, E., & Grishman, R. (1998). Description of the MENE named entity system as used in MUC-7. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia: Morgan Kaufmann Publishers, Inc.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics, 21*(4), 543–565.

Chinchor, N. (1995). Statistical significance of MUC-6 results. *Proceedings of the Sixth Message Understanding Conference (MUC-6)* (pp. 39–43). Columbia, Maryland: Morgan Kaufmann Publishers, Inc.

Chinchor, N. (1998). MUC-7 named entity task definition dry run version, version 3.5, 17 September 1997. *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (to appear). Fairfax, Virginia: Morgan Kaufmann Publishers, Inc. URL: ftp://online.muc.saic.com/NE/training/guidelines/NE.task.def.3.5.ps.

Church, K. (1988). A stochastic parts program and noun phrase parser for unrestricted text. *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, Texas.

Krupka, G. (1995). SRA: Description of the SRA system as used for MUC-6. *Proceedings of the Sixth Message Understanding Conference (MUC-6)* (pp. 221–235). Columbia, Maryland: Morgan Kaufmann Publishers, Inc.

Merchant, R., Okurowski, M., & Chinchor, N. (1996). The multilingual entity task overview. *Proceedings of the Tipster Text Program Phase II* (pp. 445–447). Vienna, Virginia: Morgan Kaufmann Publishers, Inc.

Rabiner, L.R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*.

Sundheim, B., & Chinchor, N. (1995). Named entity task definition (version 2.1). *Proceedings of the Sixth Message Understanding Conference (MUC-6)* (pp. 319–332). Columbia, Maryland: Morgan Kaufmann Publishers, Inc.

Viterbi, A.J. (1967). Error bounds for convolutional codes and an asympotically optimum decoding algorithm. *IEEE Transactions on Information Theory, IT-13*(2), 260–269.

Weischedel, R. (1995). BBN: Description of the PLUM system as used for MUC-6. *Proceedings of the Sixth Message Understanding Conference (MUC-6)* (pp. 55–69). Columbia, Maryland: Morgan Kaufmann Publishers, Inc.

Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L., & Palmucci, J. (1993). Coping with ambiguity and unknown words through probabilistic methods. *Computational Linguistics, 19*(2), 359–382.