Computational Intelligence, Volume X, Number Y, 2000

APPLYING MACHINE LEARNING FOR HIGH PERFORMANCE NAMED-ENTITY EXTRACTION

SHUMEET BALUJA VIBHU O. MITTAL RAHUL SUKTHANKAR

Just Research, 4616 Henry Street, Pittsburgh, PA 15213 &
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

This paper describes a machine learning approach to building an efficient, accurate and fast name spotting system. Finding names in free text is an important task in many text-based applications. Most previous approaches were based on hand-crafted modules encoding language and genre-specific knowledge. These approaches had at least two shortcomings: they required large amounts of time and expertise to develop, and were not easily portable to new languages and genres. This paper describes an extensible system which automatically combines weak evidence from different, easily available sources: part-of-speech tags, dictionaries, and surface-level syntactic information such as capitalization and punctuation. Individually, each piece of evidence is insufficient for robust name detection. However, the combination of evidence, through standard machine learning techniques, yields a system that achieves performance equivalent to the best existing hand-crafted approaches.

1. INTRODUCTION

Spotting named-entities in text can be an important component of tasks such as information extraction and retrieval,[1] restoration of capitalization in single-case text, and spelling-correction — to avoid accidentally "correcting" names that are mistaken as misspelled words.

Most previous research efforts in building named-entity systems relied on carefully hand-crafted rules. There are two impediments to this approach. First, most of the typical hand-coded systems are optimized to work well in very specific situations; however, both language and users' needs are likely to evolve over time. Second, it is not clear how such systems could be easily modified to reflect less well behaved texts – ones that may have misspelled words, missing case information, or foreign words/phrases [12].

This paper presents a system for named-entity extraction that is automatically trained to recognize named-entities using statistical evidence from a training set. This approach has several advantages: first, it eliminates the need for expert language-specific linguistic knowledge. To train the automated system, users only need to tag items that interest them. If the users' needs change, the system can re-learn from new data quickly. Second, system performance can be improved by increasing the amount of training data without requiring extra expert knowledge. Third, if new knowledge sources become available, they can easily be integrated into the system as additional evidence.

2. BACKGROUND AND PREVIOUS WORK

Previous work in this area has largely taken place in the context of the Message Understanding Conferences (MUCs) [9]. For MUC, the problem of finding named-entities was divided into three sub-problems ENAMEX, finding entity names (organizations, persons and locations); TIMEX, finding temporal expressions (dates and times); and NUMEX, finding numerical quantities (monetary values, percentages, etc.). However, as discussed by Palmer and Day [12], the latter two tasks are far simpler than the first. (For instance, they found that nearly all NUMEX phrases could be identified reliably

[1] A study on IR in a legal domain found a 20% improvement in precision when users could specifically search for names [18].

with a very small number of patterns.) Since ENAMEX has been suggested to be the most difficult of the three sub-tasks, this paper concentrates on name detection.

Early work on name detection was based on either: (1) hand-crafted regular expressions [2; 3; 20]; (2) extensive resources such as lists of geographic locations, people and company names, etc. [10]; or (3) sophisticated linguistic approaches based on parsing [11; 10; 8]. Such systems can be quite expensive to develop and maintain. Systems that exploit machine learning are more closely related to our work. NYMBLE [4], based on a Hidden Markov Model (HMM) [15], achieves good performance at the expense of large computational resources. ALEMBIC [1] performs named-entity extraction by learning rule sequences, but uses different feature encodings and does not parameterize the contribution of its various components.

3. KNOWLEDGE SOURCES USED

To determine whether a token is a name, our system uses weak evidence from a number of sources. The main consideration in deciding which information sources to use was the difficulty associated with creating and maintaining such sources. A secondary consideration was to keep the learned models as small as possible. The knowledge sources, encoded as a set of 29 features, are:

Word Level Features: Language- or genre-specific cues can sometimes be exploited to provide evidence for name detection (e.g., in English, names are often capitalized). The following features are used in encoding tokens and the system learns which of these correlate strongly with names: (1) *all-uppercase*, (2) *initial-caps*, (3) *all-numbers*, (4) *alphanumeric*, (5) *single-char*, (6) *single-s* (if the token is the character "s"), and (7) *single-i* (if the token is the character "I").

Individually, none of the local word-level features are very effective: the strongest individual feature is *all-caps*; it flags 474 tokens in the training set (containing 50,416 tokens, of which 3,632 are names). Of these, 449 are actually names, the rest being non-name acronyms such as "CEO" and "PC", yielding an F_1-score[2] of only 21 (Precision = 94; Recall = 12). The feature *initial-caps* is similar, flagging 5,650 words, of which 3,572 are names, leading to an F_1-score of 82 (Precision = 73; Recall = 94). Note that not all capitalized words are names, and that not all names are capitalized (e.g., "van Gogh").

For English text, capitalization, in conjunction with reliable end-of-sentence boundary detection, is a good indicator for names. However, determining sentence boundaries is difficult since common boundaries such as periods, question- and exclamation-marks can occur in many different contexts [16; 13]. While the system does not explicitly contain rules for sentence boundary analysis, by using contextual cues, it can account for many sentence boundaries.

Dictionary Look-Up: Another weak heuristic for determining whether a particular token is a name is to check whether it can be found in a dictionary. Since many names are not valid English words, this resource can identify some potential names. The dictionary used in our experiments was the standard spelling dictionary available on most UNIX systems; it contained 45,402 words, of which 6,784 were capitalized, and were discarded as names. The remaining 38,618 tokens contained multiple morphological variants of the same word (further decreasing the number of unique root forms). Finally, since a number of English names are also part of the regular vocabulary (e.g., "mark", "baker" and

[2]Performance on the name detection task is typically measured by the F_β score [19], which is a combination of the *Precision* (P) and *Recall* (R) measures used in IR. The F_β score is defined to be: $\frac{(\beta^2+1)*R*P}{\beta^2*P+R}$, where β is usually set to 1. Studies have shown that, on average, the F_1 score for *manually* finding names in text is approximately 96 [9]. In comparison, the F_1 scores for many manually crafted systems are often between 90 and 92 [10; 8; 5; 17].

"stone"), name detection using only evidence from the dictionary is not very reliable: the F_1-score for the dictionary module alone on our training set was only 64.

Part-of-Speech Tagger: Part-of-Speech (POS) tags can be used by other modules to reason about the roles and relative importance of words/tokens in various contexts. In this system, we used the Brill tagger for POS tagging[3]. Brill reports approximately 97% overall accuracy for words in the WSJ corpus for the tagger [6; 7]. Its performance is lower on the named-entity task: on our training data, the tagger obtained an F_1-score of 83 (P = 81, R = 86); consistent results are reported in [1]. The following POS tags were used as features by the machine learning component of our system: (1) *determiner*, (2) *foreign-word*, (if the token is one that the tagger has not seen), (3) *preposition*, (4) *adjective*, (5) *noun*, (6) *proper-noun*, (7) *personal-pronoun*, (8) *possessive-pronoun*, (9) *verb*, (10) *WH-pronoun* (which, what, etc.), (11) *unknown-POS*.

Punctuation: Robust name detection probably requires that the system capture contextual syntactic information. (At the very least, to disambiguate capitalization cues due to sentence boundaries.) The system learns syntactic patterns that may indicate named entities. Section 5 discusses the effects of varying the size of this contextual window. The following punctuation characters are encoded as features: (1) comma; (2) period; (3) exclamation mark; (4) question mark; (5) semi-colon; (6) colon; (7) plus or minus sign; (8) apostrophe; (9) left parenthesis; (10) right parenthesis.

4. SYSTEM ARCHITECTURE

The name spotting system consists of two components: a tokenizer and a classifier. The tokenizer converts text into a set of features based on the knowledge sources presented in the previous section. These tokens are used by the classifier, which is a decision tree constructed from the training data based on information theory (C4.5 [14]).

The tokenizer reads input text and creates tokens consisting of either words or selected punctuation marks. Each token is encoded by a set of 29 boolean features (+1 or -1) indicating the presence or absence of that feature. Features that are not used in a particular experiment are given a zero value.

The classifier combines the various sources of weak evidence about the candidate token and its context. No feature, by itself, is sufficient for robust classification. The goal of the classifier is to automatically combine all of the evidence to determine whether the candidate token is a name. The output, or target variable, is the manually-coded label identifying whether the token is a name.

Since the classifier does not explicitly model syntactic patterns in the text, the decision tree learner must induce a large number of syntactic patterns to account for varying numbers of tokens that can appear in a particular syntactic role. One approach to dealing with this problem is to collapse adjacent syntactic tokens from the same category into a single token. This compression permits the system to learn fewer patterns in the available data while maintaining (or perhaps improving) accuracy. Although computationally more expensive, as will be shown in Section 5, our experiments reveal an improvement in performance with a limited version of this method.

[3]Version 1.1, with 148 lexical rules and 283 contextual rules, trained on a Wall Street Journal (WSJ) corpus from the Linguistic Data Consortium with a lexicon of 93,696 words.

5. EXPERIMENTAL RESULTS

Our experiments were conducted using a training set of 100 randomly-selected Reuters news articles, containing 50,416 tokens, of which 44,013 were words (the rest were punctuation). The training set included 3,632 names, 1552 of which were distinct. The results reported in this section were obtained by running the system on a test set of 25 additional articles (these articles were not used for training). These test articles contained a total of 13,507 tokens, of which 11,811 were words, and 1048 were labeled by the coders as names.

Section 3 discussed baseline performance for each of the individual modules. One simple "learning" approach would be for the system to construct a list of names encountered in the training set and match candidate tokens against this list during testing. However, as pointed out in [12], this is unlikely to significantly help in name detection. Our observations confirmed this hypothesis: the test set contained 1048 names (of which 441 were unique). Of these 1048 names, only 110 names had appeared in the training set; therefore, the system cannot simply rely on a list of names built during training.

The experimental procedure was as follows: (1) the manually labeled data was divided into three sets, *training*, *validation*, and *testing*; (2) the training set was used for inducing the decision tree; (3) the validation set was used to prevent over-fitting of the data; (4) during training, the error on the validation set was tracked. When this error started to increase due to overfitting, the training was halted and the classifier was evaluated on the testing set. To ensure that idiosyncrasies in any data-set splitting did not affect our results, repeated tests were employed to accurately estimate the system's performance. Each experiment was repeated 5 times, using different parts of the data-set for training and validation. In each experiment, 80 articles were used for training, and 20 for validation. All of the results presented are measured on the performance of the network on an entirely separate testing set of 25 articles.

The first line of Table 1 shows the performance of the system with only the information for the word to be classified. The second line shows the performance when given context information; in addition to the 29 entry feature vector for the word to be classified, the decision tree is also given the feature vectors for one word before and after the word to be classified. The third and fourth lines show similar context information for 2 and 6 words before and after the word to be classified, respectively. Note that the context is taken without regard to sentence boundaries.[4] For simplicity, the number of words examined before and after the candidate token were kept the same; however, this is not a requirement for the algorithm.

We also examined the performance of using single and pair-wise combinations of knowledge sources; the results of these are shown in Table 2. Note that the dictionary features combined with the word-level features perform almost as well as the word-level features combined with the part-of-speech (POS) tags. However, the dictionary and POS tags combination does not perform as well. This suggests that the word level features contain information that is not contained in either of the other two sources.

There are several methods for understanding the importance of individual features. The first is to examine the rules that are encoded in the decision tree. Sample rules are shown in Figure 1. A second method to understand each feature's potential contribution to the classification is to examine the weights for each of the features in a simple perceptron. In this study, the perceptron is trained independently from the decision-tree, and achieves approximately the same errors rates. With the perceptron, a positive weight implies that the respective feature is positively correlated with the candidate token being a name; negative weights are negatively correlated in the same manner.

Figure 2 depicts the weights in a trained perceptron with context of 0 and 1 respectively. Sev-

[4] A token, such as a period, which may indicate a sentence boundary, is part of the context, and thus enables the system to eventually learn about rules for capitalizing the first word of a sentence.

KNOWLEDGE SOURCES	Context words(0–6)	Accuracy Recall	Prec.	F_1
ALL	0	0.932	0.899	**91.5**
ALL	1	0.941	0.923	**93.2**
ALL	2	0.942	0.931	**93.1**
ALL	6	0.937	0.924	**93.0**

TABLE 1. Using All Knowledge Sources and the Effects of Context. (Averaged over 5 runs)

Kl. Sources	# of words	Accuracy Recall	Prec.	F_1
P	0	0.855	0.814	**83.3**
P	1	0.857	0.802	**82.8**
P	2	0.854	0.818	**83.5**
P	6	0.862	0.808	**83.7**
D	0	0.910	0.483	**62.8**
D	1	0.915	0.480	**62.9**
D	2	0.911	0.489	**63.6**
D	6	0.918	0.476	**62.6**
WL	0	0.983	0.637	**77.3**
WL	1	0.981	0.640	**77.4**
WL	2	0.981	0.613	**75.4**
WL	6	0.980	0.627	**76.4**

(a) Effects of varying context.

Kl. sources	# of words	Accuracy Recall	Prec.	F_1
D & P	0	0.646	0.831	**72.6**
D & P	1	0.760	0.780	**76.9**
D & P	2	0.822	0.774	**79.7**
D & P	6	0.752	0.777	**76.4**
D & WL	0	0.639	0.929	**75.7**
D & WL	1	0.931	0.910	**92.0**
D & WL	2	0.949	0.911	**93.9**
D & WL	6	0.941	0.912	**92.6**
P & WL	0	0.906	0.912	**90.8**
P & WL	1	0.911	0.901	**90.5**
P & WL	2	0.932	0.918	**92.4**
P & WL	6	0.923	0.883	**90.2**

(b) Effects of pair-wise combinations.

TABLE 2. Performance of individual and combined knowledge sources, with different amounts of context averaged over 5 runs. P=Part of Speech, D=Dictionary, WL=Word Level Features.

eral attributes of these figures should be noticed. First, in the zero context case, the features most indicative of names are those for capitalization. As discussed in Section 3, POS tagging for proper nouns is not very reliable. This is reflected by the medium weight given to the proper noun tag by the classifier. It is worth noting that the POS-tagger's label of noun and adjective are indicative of proper names. This suggests that the proper-name detection of the tagger sometimes misses nouns and adjectives which should be labeled as names. The most salient features *against* proper names are whether the word exists in the dictionary and whether it is a single character. In the single-word context models, the classifiers learned to exploit domain-specific idiosyncrasies. Since all of training articles came from the Reuters news-wire stories, the articles always contain the token `(Reuters)`, where "Reuters" is flagged as a name in the training set. The classifier learns to tag candidate tokens with parentheses on either side as names.

Finally, it is promising that although there are many input features, the system automatically ignores the irrelevant features. When porting the system to other languages or genres, it may not be obvious which features to use. The system allows us to add many potential features from which the relevant ones are automatically selected.

As mentioned earlier, the variability in language makes the number of syntactic patterns that the system has to learn be much larger than can be learned using the limited training data that we had

```
Rule 39:
   comma-1 = +1
   word-in-dict-2 = -1
   initial-caps-2 = +1
   noun-3 = -1
 ->  named-
entity  [97.8%]

Rule 91:
   unknown-pos-1 = -1
   proper-noun-2 = +1
   all-uppercase-2 = +1
 ->  named-
entity  [97.2%]

Rule 47:
   adjective-1 = -1
   word-in-dict-2 = -1
   initial-caps-2 = +1
   verb-3 = +1
 ->  named-
entity  [95.8%]

Rule 121:
   proper-noun-2 = +1
   word-in-dict-2 = -1
   initial-caps-2 = +1
 ->  named-
entity  [95.3%]
```

FIGURE 1. Sample rules generated by C4.5, with one word context. For example, given three tokens in order, *token-1 token-2 token-3*, the context in this case are *token-1* and *token-3*; *token-2* is the token under consideration. To illustrate, Rule #47 states that if the preceding word is not an adjective, the current token is not in the dictionary, has an initial upper-case letter, and the following word is a verb, the token under consideration is a named-entity (if none of the preceding rules apply).

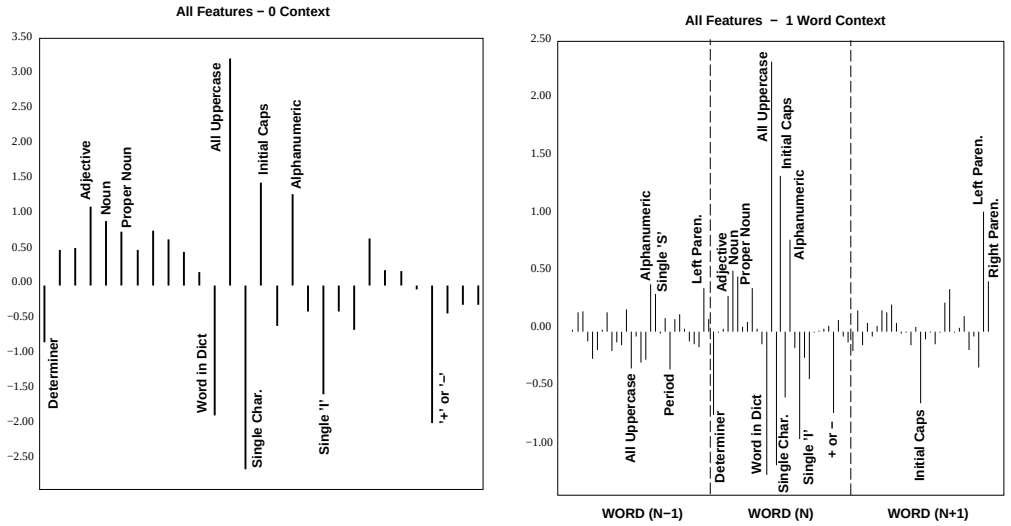

FIGURE 2. Feature weightings when using all knowledge sources; effects of zero and one word of context before and after the word to be classified are shown. Positive values signal a name, and negative values signal against a name. For clarity, only weights with large magnitudes are labeled here.

available. To increase the number of effective patterns that are learned, we collapsed adjacent similar tokens into a single token. When using this system in practice (after training), adjacent tokens that were classified as nouns by the POS tagger were collapsed into a single token. Using such a cascaded processing model resulted in improved performance, with the system's performance now beginning

NEW YORK (Reuters) - Hotel real estate investment trust Patriot American Hospitality Inc. said Tuesday it had agreed to acquire Interstate Hotels Corp, a hotel management company, in a cash and stock transaction valued at $2.1 billion, including the assumption of $785 million of Interstate debt.

Interstate's portfolio includes 40 owned hotels and resorts, primarily *upscale*, full-service facilities, leases for 90 hotels, and management-service agreements for 92 hotels.

On completion of the Interstate deal and its pending acquisitions of Wyndham Hotel Corp. and WHG Resorts and Casinos Inc., Patriot's portfolio will consist of 455 owned, leased, managed, franchised or serviced properties with about 103,000 rooms.

A definitive agreement between Patriot and Interstate values Interstate at $37.50 per share. Patriot will pay cash for 40 percent of Interstate's shares, and will exchange Patriot paired shares for the rest. Paired shares trade jointly for real estate investment trusts and their paired operating companies.

Patriot said it expects the transaction to be about 8 percent accretive to its funds from operations.

It said the agreement had been approved by the boards of Interstate and Wyndham Patriot said it did not expect the deal to delay the closing if its transaction with Wyndham, which is to close by year-end.

FIGURE 3. This example illustrates the performance of the system (with a context of one). Underlined words indicate names that were successfully detected; italicized words mark tokens that were misclassified as names; and bold words are names that were not found. See the text for details.

to match the best reported F_1 scores. With one-word context, the cascaded approach improved the F_1 score from from 93.2 to 95.2. This experiment is the first attempt of reducing the number of patterns to be learned; other methods for syntactic pattern compression are being explored.

The preceding discussion, along with the quantitative experimental results reported, has identified some of the benefits and drawbacks of our approach. We can also gain insight into the system's performance on the name detection task by examining specific failure cases. Figure 3 presents the system's output on a randomly selected news story. The underlined words are names that were successfully detected; italicized words mark tokens that were misclassified as names; and bold words are names that were missed. In this example, the system failed to find two names ("Interstate" and "Patriot"), and misclassified the word "upscale" as a name. It should be noted that the word "upscale" does not appear in the system's dictionary, while both of the names, "Interstate" and "Patriot" do. It should be noted, however, that the latter errors are not fully explained by the dictionary confusion: although the words "Interstate" and "Patriot" occur multiple times in this document, they are correctly tagged in every other instance. In these instances, the evidence gathered from sources other than the dictionary was enough to outweigh the dictionary confusion.

6. CONCLUSIONS AND FUTURE WORK

This paper presents high-performance name spotting based on machine learning. Although it achieves performance comparable to the best name detection systems, it does not rely on hand-crafted rules nor large manually created lists of names. This paper also presents an analysis of different knowledge sources, combinations of which can yield widely varying results. To design larger systems which address more complex tasks, it is important to determine which knowledge sources provide the best discrimination power, and which are redundant.

In future work, we plan to extend this system to other tasks, as well as complete the MUC task

specifications, which include name-spotting and categorizing them as people-names, location-names and organization-names. Once a potential name has been identified, there are several cues that can be exploited to determine to which of these three categories it belongs. Additionally, we plan to explore hybrid systems where our approach is used in conjunction with traditional parsing techniques.

REFERENCES

J. Aberdeen, J. Burger, David Day, Lynette Hirschman, P. Robinson, and Marc Vilain. MITRE: Description of the alembic system used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 141–155, Columbia, MD, 1995. NIST, Morgan-Kaufmann Publishers.

D. Appelt, J. Hobbs, D. Israel, and M. Tyson. FASTUS: A finite-state processor for information extraction from real-world text. In *Proceedings IJCAI-93*, 1993.

T. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, and M. Tyson. SRI international FASTUS system MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, 1995. NIST, Morgan-Kaufmann Publishers.

D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. NYMBLE: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, Washington, D.C., 1997. ACL.

A. Borkovsky. Knight-Ridder Information's Value Adding Name Finder. A Variation on the Theme of Fastus. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, 1995. NIST, Morgan-Kaufmann Publishers.

Eric Brill. Some advances in rule based part-of-speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 722–727, Seattle, WA, 1994. AAAI.

Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–566, December 1995.

R. Gaizauskas, T. Wakao, K. Humphreys, H. Cunningham, and Y. Wilks. University of Sheffield: Description of the LaSIE system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, 1995. NIST, Morgan-Kaufmann Publishers.

R. Grishman and B. Sundheim. Design of the MUC-6 evaluation. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, 1995. NIST, Morgan-Kaufmann Publishers.

L. Iwanska, M. Croll, T. Yoon, and M. Adams. Wayne state university: Description of the UNO natural language processing system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, 1995. NIST, Morgan-Kaufmann Publishers.

R. Morgan, R. Garigliano, P. Callaghan, S. Poria, M. Smith, A. Urbanowicz, R. Collingham, M. Costantino, C. Cooper, and the LOLITA Group. University of durham: Description of the LOLITA system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, 1995. NIST, Morgan-Kaufmann Publishers.

David D. Palmer and David S. Day. A statistical profile of the Named-Entity Task. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 190–193, Washington, D.C., 1997. ACL.

David D. Palmer and Marti A. Hearst. Adaptive sentence boundary disambiguation. In *Proceedings of the 1994 Conference on Applied Natural Language Processing*, Stuttgart, Germany, October 1994. ACL.

J. Ross Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann Series in Machine Learning. Morgan-Kaufmann Publishers, Menlo Park, CA, 1992.

Lawrence Rabiner. A tutorial on hidden markov models and selective applications in speech recognition. In Alex Waibel and K. F. Lee, editors, *Readings in Speech Recognition*. Morgan Kaufmann Publishers, 1993.

Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington, D.C., 1997. ACL.

B. Sundheim. Overview of results of the MUC-6 evaluation. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, 1995. NIST, Morgan-Kaufmann Publishers.

Paul Thompson and Christopher C. Dozier. Name searching and information retrieval. Available in the The Computation and Language E-Print Archive at $http://xxx.lanl.gov/abs/cmp-lg/9706017/$, June 1997.

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

Ralph Weischedel. BBN: description of the PLUM system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, 1995. NIST, Morgan-Kaufmann Publishers.