Pacific Association for Computational Linguistics

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

e-mail: {baluja,mittal,rahuls}@justresearch.com

This paper describes a machine learning approach to build an efficient, accurate and fast name spotting system. Finding names in free text is an important task in addressing real-world text-based applications. Most previous approaches have been based on carefully hand-crafted modules encoding linguistic knowledge specific to the language and document genre. Such approaches have two drawbacks: they require large amounts of time and linguistic expertise to develop, and they are not easily portable to new languages and genres. This paper describes an extensible system which automatically combines weak evidence for name extraction. This evidence is gathered from easily available sources: part-of-speech tagging, dictionary lookups, and textual 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.

Keywords:. information extraction, machine learning

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 have relied on the use of carefully hand-crafted rules. There are two impediments to their wide use. First, most of the typical hand-coded systems work well only in the specific circumstances envisioned by their designers. However, both language and users' needs are likely to evolve over time. Secondly, it is unclear what additional resources would be required to adapt to other languages, document genres, or less well-behaved texts — such as ones that may have misspelled words, missing case information, or foreign words/phrases [Palmer and Day, 1997]. Thus, it is desirable that name spotting systems should not only be accurate, but should facilitate easy user parameterization.

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. With 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 expert knowledge. Third, if new knowledge sources become available, they can easily be integrated into the system as additional evidence.

[1]A recent study on IR in a legal domain found a 20% improvement in precision when users could specifically search for names [Thompson and Dozier, 1997].

2. BACKGROUND AND PREVIOUS WORK

Previous work in this area has largely taken place in the context of the Message Understanding Conferences (MUCs) [Grishman and Sundheim, 1995]. In the case of MUC, the problem of finding named-entities was broken up 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 [Palmer and Day, 1997], the latter two tasks are much simpler than the first. In an analysis of the three tasks across five languages, they found that nearly all NUMEX phrases could be identified reliably 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 solely on name detection.

Much of the initial work on finding names was based on either: (1) carefully hand-crafted regular expressions [Appelt *et al.*, 1993; Appelt *et al.*, 1995; Weischedel, 1995]; (2) the use of extensive specialized resources such as large lists of geographic locations, people and company names, etc. [Iwanska *et al.*, 1995]; (3) highly sophisticated rule-based linguistic approaches, based on parsing [Morgan *et al.*, 1995; Iwanska *et al.*, 1995; Gaizauskas *et al.*, 1995]. Because these approaches rely on manually coded rules or large lists, such systems can be extremely expensive to develop and maintain.

While a variety of name detection algorithms have been proposed in the literature, in this paper, we mention only those that incorporate a strong machine learning component. The best known of these systems is NYMBLE [Bikel *et al.*, 1997], a statistical system based on a Hidden Markov Model (HMM) [Rabiner, 1993]. NYMBLE is reported to have an F_1 score of 93. While NYMBLE's approach is effective, it requires large computational resources. Another system using machine learning techniques with similar, but slightly lower performance, is ALEMBIC, which relies on rule sequences [Aberdeen *et al.*, 1995].

3. KNOWLEDGE SOURCES USED

To determine whether a token is a name, the system uses evidence gathered from a variety of sources. The main consideration in deciding which information sources to use is the difficulty associated with creating and maintaining the appropriate resources. For example, large, manually generated lists of people and business names are tedious to compile and require large amounts of maintenance. A second consideration is that the learned model must be easily adaptable to different genres of writing. For example, news-wire documents display very different characteristics from business or informal letters. Rapid adaptation to different genres requires that the model be easy to re-train, and that extensive genre-specific knowledge sources are not over-used. A final, pragmatic, choice was to keep the learned models extremely small. The knowledge sources that we use can be broadly divided into four categories. The 29 features derived from these knowledge sources are discussed in subsections below.

3.1. Word Level Features

Language- or genre-specific cues can sometimes be exploited to provide evidence for name detection. For instance, in English, names typically begin with a capital letter. Not only are these cues computationally inexpensive, as shown in Section 5, they can be surprisingly reliable. The following word-level features were chosen as potentially useful sources of evidence. The system automatically learns which of these features 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 (which consists of a total of 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-score[2] of only 21 (Precision = 94; Recall = 12). *initial-caps* is similar, flagging 5,650 words, of which 3,572 are names, leading to an F-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”). This problem can be particularly acute in some foreign languages such as Chinese, which does not have capitalization, in Spanish where location names are often not capitalized, and in German, where all nouns are capitalized.

For English text, capitalization, in conjunction with reliable end-of-sentence boundary detection, is an excellent indicator for names. Unfortunately, as discussed in [Reynar and Ratnaparkhi, 1997; Palmer and Hearst, 1994], determining sentence boundaries is difficult since common boundaries such as periods, question- and exclamation-marks can occur in many different contexts. For example, these punctuation marks, when used in quotations or inside parentheses do not mark the end of a sentence. Additionally, periods are commonly used in abbreviations, as decimal points, and in ellipses. While the system does not explicitly contain rules for sentence boundary analysis, as we will show, it learns sufficient contextual cues to account for most sentence boundaries.

3.2. Dictionary Look-Up

A simple heuristic for determining whether a particular token is a name is to check whether it can be found in a dictionary. Since most names are not valid English words, this approach can help identify potential names. The dictionary used in this study contained 45,402 words[3]. Of these words, 6,784 had their initial letters 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 “stone”), name detection using only evidence from the dictionary is not very reliable: the F-score for the dictionary module alone on our training set was only 64.

3.3. 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[4]. Brill reports approximately 97% to 98% overall accuracy for words in the WSJ corpus for the tagger [Brill, 1994; Brill, 1995]. Its performance is lower on the named-entity task. On our training data, the tagger obtained an F-score of only 83 (P = 81,

[2]Performance on the name detection task is typically measured by the F_β score [van Rijsbergen, 1979], 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 [Grishman and Sundheim, 1995]. In comparison, the F_1 scores for many manually crafted systems are often between 90 and 92 [Iwanska *et al.*, 1995; Gaizauskas *et al.*, 1995; Borkovsky, 1995; Sundheim, 1995].

[3]The dictionary used in the system was the standard spelling dictionary available on most UNIX systems.

[4]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.

R = 86); Aberdeen *et al.* [Aberdeen *et al.*, 1995] report consistent results. 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*.

3.4. Punctuation

For robust determination of name or not-name, our system must be able to capture syntactic information surrounding the word to be classified. As mentioned earlier, contextual information is necessary to disambiguate capitalization cues that occur due to sentence boundaries. Context around the candidate token includes surrounding punctuation. The system's classifier learns to exploit syntactic regularities automatically from the training data. Section 5 discusses the effects of varying the size of this contextual window. We consider the following punctuation marks: (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 are used by the classifier, a decision tree constructed from the training data based on information theory (C4.5 [Quinlan, 1992]).

4.1. Tokenizer

The tokenizer reads free-form text and creates tokens consisting of either words or selected punctuation marks. A feature vector of 29 elements is calculated for each such token, based on the knowledge sources described in Section 3. All of the features, when used, encode binary variables, (i.e., *initial-caps* is set to +1 if the token begins with a capital letter, and -1 otherwise). As described in Section 5, some of the experiments do not use all of the knowledge sources. If a knowledge source was not used, all of its features were set to 0.

Since the classifier does not explicitly model syntactic patterns in the text, the decision tree learner must necessarily induce a large number of syntactic patterns to take into account the variations in the numbers of tokens that can appear in a particular syntactic role. One approach to dealing with this problem would be to collapse adjacent syntactic tokens from the same category into one single token. This would present the learning system with far fewer patterns to learn and potentially improve results.[5]

4.2. Decision Tree Classifier

The classifier processes 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. Perhaps the

[5]As we will discuss in Section 5, we tried a limited version of this method of collapsing patterns and did indeed see a small improvement in performance. However, this comes about at the expense of processing speed, and will not be discussed at length here.

```
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%]

Rule 82:
          determiner-1 = -1
          adjective-1 = -1
          word-in-dict-1 = +1
          proper-noun-2 = +1
          word-in-dict-2 = -1
          initial-caps-2 = -1
          alphanumeric-2 = -1
          comma-3 = -1
       -> named-entity  [94.4%]

Rule 60:
          proper-noun-1 = +1
          proper-noun-2 = -1
          initial-caps-2 = +1
       -> named-entity  [93.9%]

Rule 66:
          all-uppercase-2 = +1
          single-char-2 = -1
          colon-3 = -1
          right-paren-3 = -1
       -> named-entity  [93.7%]

Rule 118:
          period-1 = -1
          question-1 = -1
          proper-noun-2 = +1
          initial-caps-2 = +1
          single-i-3 = -1
       -> named-entity  [92.8%]

Rule 58:
          word-in-dict-1 = +1
          initial-caps-2 = +1
       -> named-entity  [89.3%]

Rule 34:
          proper-noun-1 = +1
          word-in-dict-1 = +1
          all-numbers-2 = +1
       -> named-entity  [84.3%]

Rule 30:
          word-in-dict-1 = +1
          initial-caps-1 = +1
          preposition-2 = +1
          word-in-dict-3 = +1
          initial-caps-3 = +1
       -> named-entity  [75.9%]
```

Figure 1. Sample rules for spotting named-entities generated by C4.5. The context considered here is one token to the left and one token to the right. 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).

simplest approach is to treat these features as inputs to a statistical regression or machine learning procedure. The output, or target variable, is the manually-coded label identifying whether the token is a name. Figure 1 shows some of the rules that C4.5 came up with based on our data set.

5. EXPERIMENTAL RESULTS

All of the experiments reported here 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, with 1552 distinct names. The results reported in this section were obtained by running the system on a test set of 25 additional articles (these articles had never been seen by the system). 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 [Palmer and Day, 1997], 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.

In this section, we present some of the experimental results with our system. 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) when the validation set error was minimized, training was stopped, and the results were measured on the testing set. To ensure that idiosyncrasies in any data-set splitting did not affect our results, repeated tests are required to accurately estimate the system's performance; hence, 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.

In the first set of three experiments, we examined how the dictionary, part-of-speech and word-level knowledge sources perform when used independently of each other. (Note that the punctuation knowledge source is always used in these experiments.) We also examine the effect of context. The first line of Table 1 shows the performance of the part-of-speech tagger with only the part-of-speech information for the word to be classified. The second line shows the performance when given context information; the part-of-speech information for one word before and after the word to be classified is given. 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.[6] 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. The remainder of Table 1 examines the performance of using the part-of-speech tagger, the dictionary variables and the word-level features independently. Note that although none of these performs well independently, their performance improves when context is increased.

[6] A token, such as a period, which may, or may not, 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.

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

KNOWLEDGE SOURCES	Context words	Accuracy Recall	Prec.	F_1
POS	0	0.855	0.814	**83.3**
POS	1	0.857	0.802	**82.8**
POS	2	0.854	0.818	**83.5**
POS	6	0.862	0.808	**83.7**
Dict.	0	0.910	0.483	**62.8**
Dict.	1	0.915	0.480	**62.9**
Dict.	2	0.911	0.489	**63.6**
Dict.	6	0.918	0.476	**62.6**
Word-Lev.	0	0.983	0.637	**77.3**
Word-Lev.	1	0.981	0.640	**77.4**
Word-Lev.	2	0.981	0.613	**75.4**
Word-Lev.	6	0.980	0.627	**76.4**

In the next set of three experiments, we examined all the pair-wise combinations of the knowledge sources. Note that the dictionary features combined with the word-level features performs almost as well as the word-level features combined with the part-of-speech tagging. However, the dictionary and part-of-speech tagging do not perform as well. This suggests that the word level features contain information that is not contained in either of the other two sources. These results are summarized in Table 2.

The final experiment uses all of the knowledge sources. As can be seen by comparing the results shown in Tables 2 and 3, there is little difference between the performance of a system which uses only the part-of-speech tagger and the word-level features, and a system which uses these knowledge sources in addition to the dictionary.

To better understand how each individual feature affects the performance of the classifier, we can examine the weights for each of the features using linear regression.[7] The magnitude of these weights indicate how much importance is given to each input. 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 [8].

Figures 2 and 3 depict the weight values in a trained perceptron with context of 0 and 1 respectively. Several attributes of these figures should be noticed. First, in the no context case, the features which are most indicative of names are those which account for capitalization. As demonstrated in Section 3 the POS tagging for proper nouns is not very reliable. This is reflected by the only medium weight given to the proper noun tag by the classifier. Also notice that the POS-tagger's label of noun and adjective are indicative of proper names for the system. This suggests that the proper-name detection of the tagger

[7]In other experiments not reported here, we also trained a perceptron using the same data set and obtained similar results to those obtained using decision trees. The relative weights obtained from the linear regression as indicated by the trained perceptron are useful for gaining an insight into the problem.

[8]Note that the graphs are shown for training runs do not use a bias unit [Hertz *et al.*, 1991].

TABLE 2. Performance of Combining Knowledge Sources and the Effects of Context. (Averaged over 5 runs)

KNOWLEDGE SOURCES	Context words(0–6)	Accuracy Recall	Prec.	F_1
Dict. & POS	0	0.646	0.831	**72.6**
Dict. & POS	1	0.760	0.780	**76.9**
Dict. & POS	2	0.822	0.774	**79.7**
Dict. & POS	6	0.752	0.777	**76.4**
Dict. & Word-Lev.	0	0.639	0.929	**75.7**
Dict. & Word-Lev.	1	0.931	0.910	**92.0**
Dict. & Word-Lev.	2	0.949	0.911	**93.9**
Dict. & Word-Lev.	6	0.941	0.912	**92.6**
POS & Word-Lev.	0	0.906	0.912	**90.8**
POS & Word-Lev.	1	0.911	0.901	**90.5**
POS & Word-Lev.	2	0.932	0.918	**92.4**
POS & Word-Lev.	6	0.923	0.883	**90.2**

TABLE 3. Using All Knowledge Sources and the Effects of Context. (Averaged over 5 runs)

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**

sometimes misses nouns and adjectives which should be labeled as names. The largest features against proper names are whether the word exists in the dictionary and whether it is a single character. It is also interesting to note the context that it is developed in the 1 token context case. Since all of training articles came from the Reuters news-wire stories, the classifiers learned to exploit domain-specific idiosyncrasies. For example, the news 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. See Figure 3 for the weights on the features, and Figure 4 for an example news article.

Finally, it is promising to note 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 very limited amount of training data we had available. In order to learn the syntactic context of greatest

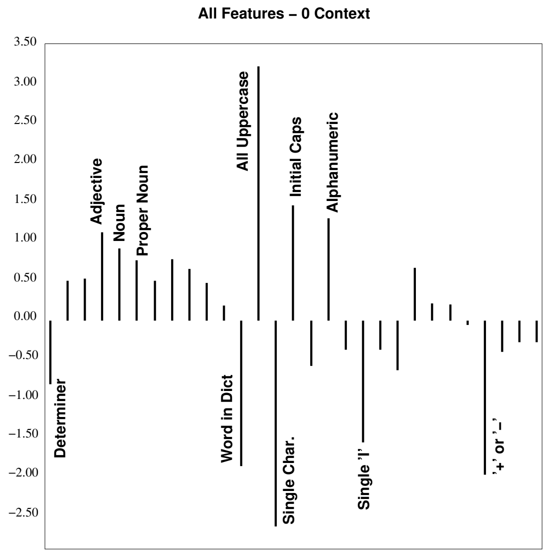

Figure 2. Feature weightings when using all knowledge sources, and no context (All-0). Positive values signal a name, and negative values signal against a name. For clarity, only weights with large magnitudes are labeled here.

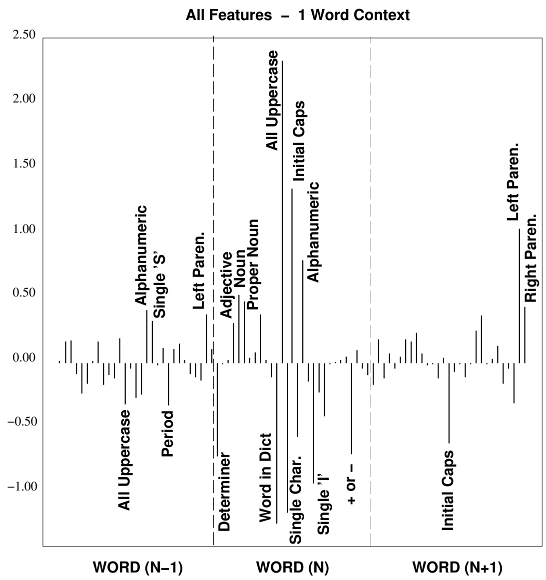

Figure 3. Feature weightings when using all knowledge sources, and 1 word of context before and after the word to be classified (All-1).

TABLE 4. A Cascaded Processing Model Seems to Give Improved Performance.

Algorithm	CONTEXT			
	0	1	2	6
One-Pass	91.50	93.20	93.70	93.05
Cascaded	92.06	95.22	95.00	94.75

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 4. This example illustrates some of the mistakes that the system (with a context of one) can make on difficult news stories. 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.

concern here, that surrounding the placement of named-entities in text, we carried out some preliminary experiments of folding multiple named tokens into a single one. In training, sequences of adjacent named entities were collapsed into a single token. In testing, 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 figures, with the system's performance now beginning to match the best reported F_1 scores, either by machine or by humans. Table 4 shows these results. It should be emphasized that these are preliminary, and we are looking at ways in which syntactic patterns can be better compressed, for both training and usage.

The preceding discussion, along with the experimental results reported in Tables 1, 2, and 3, 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 4 presents the system's output on a difficult 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. However, the latter errors are not simply caused by dictionary confusion: although the words "Interstate" and "Patriot" occur multiple times in this document, they are correctly tagged in every other instance. Some insights into this failure are given by the feature weightings in Figure 3: when the system is restricted to a context of one word, it learns that capitalized tokens are likely to be names — but not when these tokens also appear in the dictionary and are preceded by a period (rudimentary end-of-sentence detection).

6. APPLICATIONS: A BROWSING INTERFACE TO THE WWW

As an immediate application of this technology, we implemented a specialized interface for reading news articles on the WWW. The name spotting technology was built into a proxy which filtered the textual data of all documents requested by a browser. In addition, one of our colleagues implemented a system which, when given a named entity, attempts to find the "official" home page of that entity.[9] This application also highlighted another important aspect of our approach. Since our system does not use a detailed parse of the text, it is very fast. This is an important factor in user acceptance; the fact that the proxy does not impose any noticeable delays in serving pages is very important. A screen shot of news article served using the proxy is shown in Figure 6. As can be seen, almost all the named-entities in the document were correctly identified, and if a "home page" could be found for them, a hyperlink to that page was generated and inserted in the text *automatically*. We believe that a machine learning approach such as the one described here can be, and will be, used in the future to allow users to interactively mark text and images of interest, and the browsers will automatically learn to filter incoming pages and highlight portions of interest to the individual reader.

7. CONCLUSIONS AND FUTURE WORK

This paper presents a high-performance name extraction system based on machine learning techniques. Although it achieves performance comparable to the best name detection systems, the system does not rely on hand-crafted rules or large manually created databases of names. This paper also presents an analysis of the value of different knowledge sources. Different combinations of knowledge sources 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 hope to extend this system to other tasks. For example, we plan to fulfill the complete MUC task specifications, which includes name-spotting and categorizing the names as people-names, location-names and organization-names. Once a potential name

[9]This is similar in functionality to pages such as *ahoy* and others that attempt to do the same thing.

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.

Acknowledgments

```
The authors would like to thank Tony Brusseau for his work on an initial
implementation of the system, termed J-Name, which was presented in
Tokushima, Japan.  Scott Fahlman has also contributed ideas for applications.
The views and conclusions of this document do not reflect official policies
of Justsystem Pittsburgh Research Center, Just Research, or
Justsystem Corporation.  Mark Kantrowitz designed and implemented the
official-home page finder; the proxy that combined the name spotting
technology and the home page finder was implemented by Michele Banko.
```

FIGURE 5. This figure shows the acknowledgments of this paper, as processed by the system. All of the names are correctly identified.

REFERENCES

[Aberdeen *et al.*, 1995] 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.

[Appelt *et al.*, 1993] 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.

[Appelt *et al.*, 1995] 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.

[Bikel *et al.*, 1997] 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.

[Borkovsky, 1995] 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.

[Brill, 1994] 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.

[Brill, 1995] 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.

[Gaizauskas *et al.*, 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.

[Grishman and Sundheim, 1995] 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.

[Hertz *et al.*, 1991] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1991.

[Iwanska *et al.*, 1995] 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.

[Morgan *et al.*, 1995] 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.

[Palmer and Day, 1997] 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.

[Palmer and Hearst, 1994] 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.

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

[Rabiner, 1993] 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.

[Reynar and Ratnaparkhi, 1997] 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.

[Sundheim, 1995] 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.

[Thompson and Dozier, 1997] 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.

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

[Weischedel, 1995] 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.

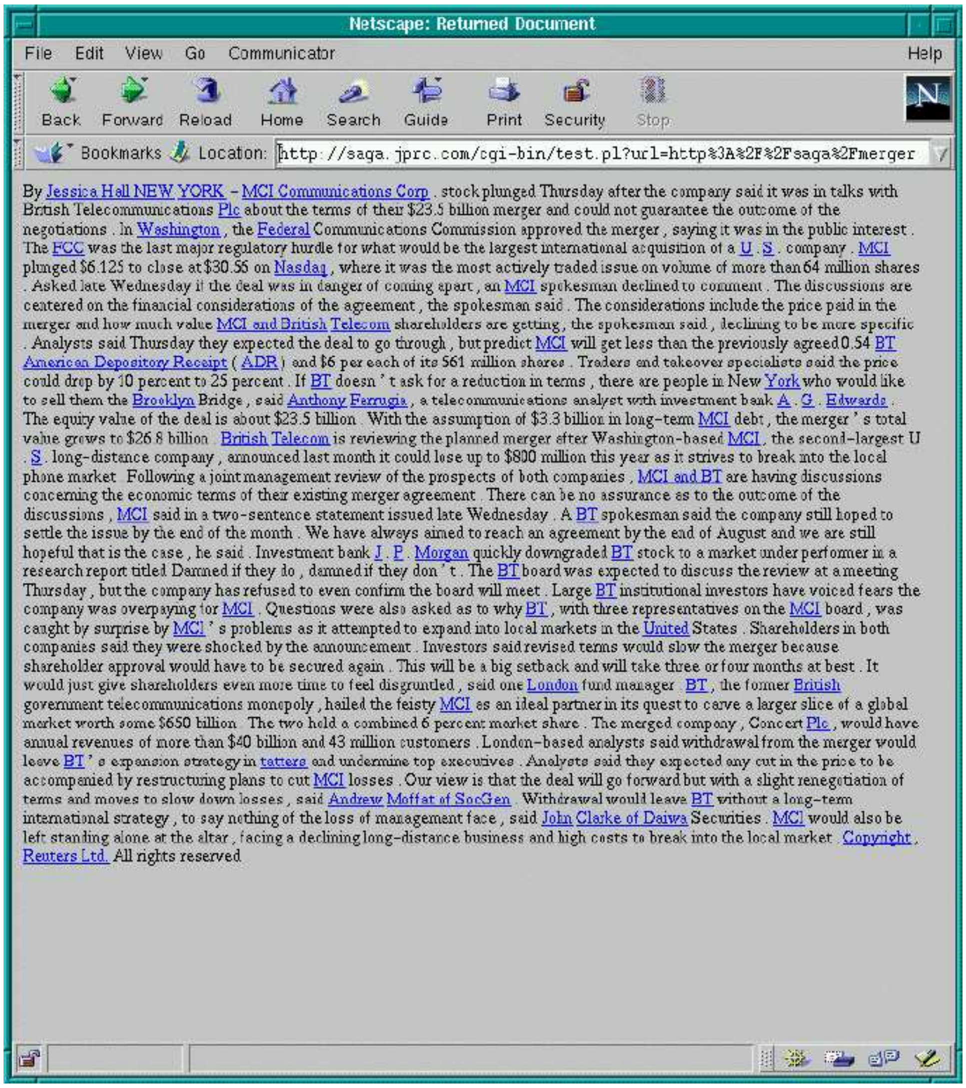

FIGURE 6. A screen shot of the browser interface built using the name spotting technology described in this paper (this one implemented using a context window of 1 word for speed).