Fine Grained Classification of Named Entities

Michael Fleischman and Eduard Hovy
USC Information Science Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
U.S.A.
{fleisch, hovy} @ISI.edu

Abstract

While Named Entity extraction is useful in many natural language applications, the coarse categories that most NE extractors work with prove insufficient for complex applications such as Question Answering and Ontology generation. We examine one coarse category of named entities, persons, and describe a method for automatically classifying person instances into eight finer-grained subcategories. We present a supervised learning method that considers the local context surrounding the entity as well as more global semantic information derived from topic signatures and WordNet. We reinforce this method with an algorithm that takes advantage of the presence of entities in multiple contexts.

1. Introduction

There has been much interest in the recent past concerning automated categorization of named entities in text. Recent advances have made some systems (such as BBN's IdentiFinder (Bikel, 1999)) very successful when classifying named entities into broad categories, such as person, organization, and location. While the accurate classification of general named entities is useful in many areas of natural language research, more fine-grained categorizations would be of particular value in areas such as Question Answering, information retrieval, and the automated construction of ontologies.

The research presented here focuses on the subcategorization of person names, which extends research on the subcategorization of location names (Fleischman, 2001). While locations can often be classified based solely on the words that surround the instance, person names are often more challenging because classification relies on much deeper semantic intuitions gained from the surrounding text. Further, unlike the case with location names, exhaustive lists of person names by category do not exist and cannot be relied upon for training and test set generation. Finally, the domain of person names presents a challenge because the same individual (e.g., "Ronald Reagan") is often represented differently at different points in the same text (e.g., "Mr. Reagan", "Reagan", etc.).

The subcategorization of person names is not a trivial task for humans either, as the examples below illustrate. Here, names of persons have been encrypted using a simple substitution cipher. The names are of only three subtypes: politician, businessperson, and entertainer, yet prove remarkably difficult to classify based upon the context of the sentence.

1. Unfortunately, *Mocpm*_____ and his immediate family did not cooperate in the making of the film .
2. "The idea that they'd introduce *Npn Fuasm*______ into that is amazing ,"he said.
3. "It's dangerous to be right when government is wrong ," *Lrsyomh*______ told reporters

1. *Mocpm* = Nixon: politician
2. *Npn Fuasm* = Bob Dylan: entertainer
3. *Lrsyomh* = Keating: businessperson

In this work we examine how different features and learning algorithms can be employed to automatically subcategorize person names in text. In doing this we address how to inject semantic information into the feature space, how to automatically generate training sets for use with supervised learning algorithms, and how to handle orthographic inconsistencies between instances of the same person.

2. Data Set Generation

A large corpus of person instances was compiled from a TREC9 database consisting of articles from the Associated Press and the Wall Street Journal. Data was word tokenized, stemmed using the Porter stemming algorithm (Porter, 1980), part of speech tagged using Brill's tagger (Brill, 1994), and named entity tagged using BBN's IdentiFinder (Bikel, 1999). Person instances were classified into one of eight categories: athlete, politician/government, clergy, businessperson, entertainer/artist, lawyer, doctor/scientist, and police. These eight categories were chosen because of their high frequency in the corpus and also because of their usefulness in applications such as Question Answering. A training set of roughly 25,000 person instances was then created using a partially automated classification system.

In generating the training data automatically we first attempted to use the simple tagging method described for location names in (Fleischman, 2001). This method involved collecting lists of instances of each category from the Internet and using those lists to classify person names found by IdentiFinder. Although robust with location names, this method proved inadequate with persons (in a sample of 300, over 25% of the instances were found to be incorrect). This was due to the fact that the same name will often refer to multiple individuals (e.g., "Paul Simon" refers to a politician, an entertainer, and Belgian scientist).

In order to avoid this problem we implemented a simple bootstrapping procedure in which a seed data set of 100 instances of each of the eight categories was hand tagged and used to generate a decision list classifier using the C4.5 algorithm (Quinlan, 1993) with the word frequency and topic signature features described below. This simple classifier was then run over a large corpus and classifications with a confidence score above a 90% threshold were collected. These confident instances were then compared to the lists collected from the Internet, and, only if there was agreement between the two sources, were the instances included in the final training set. This procedure produced a large training set with very few misclassified instances (over 99% of the instances in a sample of 300 were found to be correct). A validation set of 1000 instances from this set was then hand tagged to assure proper classification.

A consequence of using this method for data generation is that the training set created is not a random sample of person instances in the real world. Rather, the training set is highly skewed, including only those instances that are both easy enough to classify using a simple classifier and common enough to be included in lists found on the Internet. To examine the generalizability of classifiers trained on such data, a held out data set of 1300 instances, also from the AP and WSJ, was collected and hand tagged.

3. Features

3.1 Word Frequency Features

Each instance in the text is paired with a set of features that represents how often the words surrounding the target instance occur with a specific sub-categorization in the training set. For example, in example sentence 2 in the introduction, the word "introduce" occurs immediately before the person instance. The feature set describing this instance would thus include eight different features; each denoting the frequency with which "introduce" occurred in the training set immediately preceding an instance of a politician, a businessperson, an entertainer, etc. The feature set includes these eight different frequencies for 10 distinct word positions (totaling 80 features per instance). The positions used include the three individual words before the occurrence of the instance, the three individual words after the instance, the two-word bigrams immediately before and after the instance, and the three-word trigrams immediately before and after the instance (see Figure 1).

#	Position	N-gram	Category	Freq.
1	previous unigram	"*introduce*"	politician	3
2	previous unigram	"*introduce*"	entertainer	43
3	following bigram	"*into that*"	politician	2
4	following bigram	"*into that*"	business	0

Figure 1. Subset of word frequency features for instance in example 2, above. Shows the frequency with which an n-gram appears in the training data in a specific position relative to instances of a specific category.

These word frequency features provide information similar to the binary word features that are often used in text categorization (Yang, 1997) with only a fraction of the dimensionality. Such reduced dimensionality feature sets can be preferable when classifying very small texts (Fleischman, in preparation).

3.2 Topic Signature Features

Inspection of the data made clear the need for semantic information during classification. We therefore created features that use topic signatures for each of the person subcategories. A topic signature, as described in (Lin and Hovy, 2000), is a list of terms that can be used to signal the membership of a text in the relevant topic or category. Each term in a text is given a topic signature score that indicates its ability to signal that the text is in a relevant category (the higher the score, the more that term is indicative of that category). The topic signatures are automatically generated for each specific term by computing the *likelihood ratio* (λ-score) between two hypotheses (Dunning, 1993). The first hypothesis (h1) is that the probability (p1) that the text is in the relevant category, given a specific term, is equivalent to the probability (p2) that the text is in the relevant category, given *any* other term (h1: p1=p2). The second hypothesis (h2) is that these two probabilities are not equivalent, and that p1 is *much greater* than p2 (h2: p1>>p2). The calculation of this likelihood ratio [-2logL(h1)/L(h2)] for each feature and for each category gives a list of all the terms in a document set with scores indicating how much the presence of that term in a specific document indicates that the document is in a specific category.

Politician		**Entertainer**	
Word	λ-score	Word	λ-score
campaign	3457.049	Star	3283.872
republican	1969.707	Actor	2478.675
budget	140.292	Budget	17.312
bigot	2.577	Sexist	3.874

Figure 2. Subset of topic signatures generated from training set for two categories.

In creating topic signature features for the subcategorization of persons, we created a database of topic signatures generated from the training set (see Figure 2).[1] Each sentence from the training set was treated as a unique document, and the classification of the instance contained in that sentence was treated as the relevant topic. We implemented the algorithm described in (Lin and Hovy, 2000) with the addition of a cutoff, such that the topic signatures for a term are only included if the p1/p2 for that term is greater than the mean p1/p2 over all terms. This modification was made to ensure the assumption that p1 is *much greater* than p2. A weighted sum was then computed for each of the eight person subcategories according to the formula below:

[1] To avoid noise, we used only those sentences in which each person instance was of the same category.

$$\textbf{Topic Sig Score}_{\textbf{Type}} = \Sigma^N [\ \lambda\text{-score of word}_{n,Type} / (\text{distance from instance})^2]$$

where N is the number of words in the sentence, *λ-score of word$_{n,Type}$* is the topic signature score of word *n* for topic *Type*, and *distance from instance* is the number of words away from the instance that word *n* is. These topic signature scores are calculated for each of the eight subcategories.

These eight topic signature features convey semantic information about the overall context in which each instance exists. The topic signature scores are weighted according to the inverse square of their distance under the (not always true) assumption that the farther away a word is from an instance, the less information it bears on classification. This weighting is particularly important when instances of different categories occur in the same sentence (e.g., "…of those donating to Bush's campaign was actor Arnold Schwarzenegger…").

3.3 WordNet Features

A natural limitation of the topic signature features is their inability to give weight to related and synonymous terms that do not appear in the training data. To address this limitation, we took advantage of the online resource WordNet (Fellbaum, 1998). The WordNet hypernym tree was expanded for each word surrounding the instance and each word in the tree was given a score based on the topic signature database generated from the training data. The scores were then weighted by the inverse of their height in the tree and then summed together, similarly to the procedure in (Resnik, 1993). These sums are computed for each word surrounding the instance, and are summed according to the weighting process described above. This produces a distinct WordNet feature for each of the eight classes and is described by the equation below:

$$\textbf{WordNet Score}_{\textbf{Type}} = \Sigma^N[\Sigma^M \lambda\text{-score of word}_{m,Type}/(\text{depth of word}_m \text{ in WordNet})] / (\text{distance from instance})^2$$

where the variables are as above and *M* is the number of words in the WordNet hypernym tree. These WordNet features supplement the coverage of the topic signatures generated from the training data by including synonyms that may not have existed in that data set. Further, the features include information gained from the hypernyms themselves (e.g., the hypernym of "Congress" is "legislature"). These final hypernym scores are weighted by the inverse of their height in the tree to reduce the effect of concepts that may be too general (e.g., at the top of the hypernym tree for "Congress" is "group"). In order to avoid noise due to inappropriate word senses, we only used data from senses that matched the part of speech. These eight WordNet features add to the above features for a total of 96 features.

4. Methods

4.1 Experiment 1: Held out data

To examine the generalizability of classifiers trained on the automatically generated data, a C4.5 decision tree classifier (Quinlan, 1993) was trained and tested on the held out test set described above.

Initial results revealed that, due to differing contexts, instances of the same name in a single text would often be classified into different subcategories. To deal with this problem, we augmented the classifier with another program, MemRun, which standardizes the subcategorization of instances based on their most frequent classification. Developed and tested in (Fleischman, 2001), MemRun is based upon the hypothesis that by looking at all the classifications an instance has received throughout the test set, an "average" sub-categorization can be computed that offers a better guess than a low confidence individual classification.

MemRun operates in two rounds. In the first round, each instance of the test set is evaluated using the decision tree, and a classification hypothesis is generated. If the confidence level of this hypothesis is above a certain threshold (THRESH 1), then the hypothesis is entered into the temporary database (see Figure 3) along with the degree of confidence of that hypothesis, and the number of times that hypothesis has been received.

Because subsequent occurrences of person instances frequently differ orthographically from their initial occurrence (e.g., "George Bush" followed by "Bush") a simple algorithm was devised for surface reference disambiguation. The algorithm keeps a record of initial full name usages of all person instances in a text. When partial references to the instance are later encountered in the text, as determined by simple regular expression matching, they are entered into the MemRun database as further occurrences of the original instance. This record of full name references is cleared after a text is examined to avoid possible instance confusions (e.g., "George W. Bush" and "George Bush Sr."). This simple algorithm operates on the assumption that partial references to individuals with the same last name *in the same text* will not occur due to human authors' desire to avoid any possible confusion.[2] When all of the instances in the data set are examined, the round is complete.

In MemRun's second round, the data set is reexamined, and hypothesis classifications are again produced. If the confidence of one of these hypotheses is below a second threshold (THRESH 2), then the hypothesis is ignored and the database value is used.[3] In this experiment, the entries in the database are compared and the most frequent entry (i.e., the max classification based on confidence level multiplied by the increment) is returned. When all instances have been again examined, the round is complete.

Instance	Class	Confidence	Occur
George Bush	Politician	97.5%	4
	Business	83.4%	1
Dana Carvey	Entertainer	92.4%	7
	Politician	72.1%	2

Figure 3. MemRun database for Decision Tree classifier

4.2 Experiment 2: Learning Algorithms

Having examined the generalizability when using automatically generated training data, we turn to the question of appropriate learning algorithms for the task. We chose to examine five different learning algorithms. Along with C4.5, we examined a feed-forward neural network with 50 hidden units, a k-Nearest Neighbors implementation (k=1) (Witten & Frank, 1999), a Support Vector Machine implementation using a linear kernel (Witten & Frank, 1999), and a naïve Bayes classifier using discretized attributes and

[2] This algorithm does not address definite descriptions and pronominal references because they are not classified by IdentiFinder as people names, and thus are not marked for fine-grained classification in the test set.

[3] The ability of the algorithm to ignore the database's suggestion in the second round allows instances with the same name (e.g., "Paul Simon") to receive different classifications in different contexts.

with feature subset selection (Kohavi & Sommerfield, 1996). For each classifier, comparisons were based on results from the validation set (~1000 instances) described above.

4.3 Experiment 3: Feature sets

To examine the effectiveness of the individual types of features, a C4.5 decision tree classifier (Quinlan, 1993) was trained on the 25,000 instance data set described above using all possible combinations of the three feature sets. The performance was ascertained on the validation set described above.

5. Results

5.1 Experiment 1: Held out data

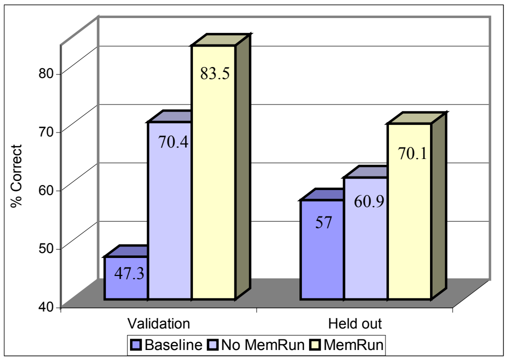

Figure 4. Results of classifier on validation set and held out data. Results compare baseline of always choosing most probable class with C4.5 classifier both with and without MemRun.

The results of the classifier on both the validation set and the held out test set can be seen in Figure 4. The results are presented for a classifier trained using the C4.5 algorithm both with and without MemRun (THRESH1=85, THRESH2=98). Also shown is the baseline score for each test set computed by always choosing the most frequent classification (Politician for both).

It is clear from the figure that the classifiers for both test sets and for both conditions performed better than baseline. Also clear is that the MemRun algorithm significantly improves performance on both the validation and held out test sets.

Figure 4 further shows a large discrepancy between the performance of the classifier on the two data sets. Expectedly, the validation set is classified more easily both with and without MemRun. The size of the discrepancy is a function of how different the distribution of the training set is from the true distribution of person instances in the world. While this discrepancy is undeniable, it is interesting to note how well the classifier generalizes given the very biased sample upon which it was trained.

5.2 Experiment 2: Learning Algorithms

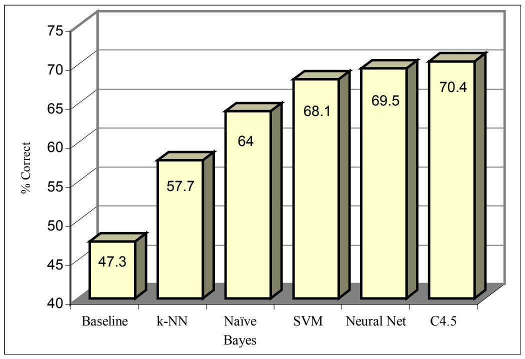

Figure 5. Comparison of different learning algorithms on a validation set. Learners include: k-Nearest Neighbors, Naïve Bayes, support vector machine, neural network, and C4.5 decision tree.

Figure 5 shows the results of comparing different machine learning strategies. It is clear from the figure that all the algorithms perform better than the baseline score, while the C4.5 algorithm performs the best. This is not surprising as decision trees combine powerful aspects of non-linear separation and feature selection.

Interestingly, however, there is no clear relationship between performance and the theoretical foundations of the classifier. Although the two top performers (decision tree and the neural network) are both non-linear classifiers, the linear SVM outperforms the non-linear k-Nearest Neighbors. This must, however, be taken with a grain of salt, as little was done to optimize either the k-NN or SVM implementation.

Another interesting finding in recent work is an apparent relationship between classifier type and performance on held out data. While the non-parametric learners, i.e. C4.5 and k-NN, are fairly robust to generalization, the parametric learners, i.e. Naïve Bayes and SVM, perform significantly worse on the new distribution. In future work, we intend to examine further this possible relationship.

5.3 Experiment 3: Feature sets

The results of the feature set experiment can be seen in figure 6. Results are shown for the validation set using all combinations of the three feature sets. A baseline measure of always classifying the most frequent category (Politician) is also displayed.

It is clear that each of the single feature sets (frequency features, topic signature features, and WordNet features) is sufficient to outperform the baseline. Interestingly, topic signature features outperform WordNet features, even though they are similar in form. This suggests that the WordNet features are noisy and may contain too much generality. It may be more appropriate to use a cutoff, such that only the concepts two levels above the term are examined. Another source of noise comes from words with multiple senses. Although our method uses only word senses of the appropriate part of speech, WordNet still often provides many different possible senses.

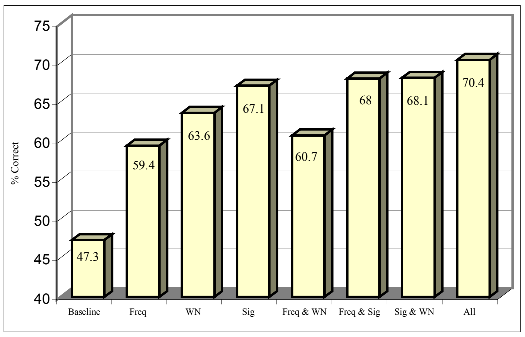

Figure 6. Results of using different combinations of feature sets. Results shown on validation set using C4.5 classifier without MemRun.

Also of interest is the effect of combining any two feature sets. While using topic signatures and either word frequencies or WordNet features improves performance by a small amount, combining frequency and WordNet scores results in performance worse than WordNet alone. This suggests over fitting of the training data and may be due to the noise in the WordNet features.

It is clear, however, that the combination of all three features provides considerable improvement in performance over any of the individual features. In future work we will examine how ensemble learning (Hastie, 2001) might be used to capitalize further on these qualitatively different feature sets.

6. Related Work

While much research has gone into the coarse categorization of named entities, we are not aware of much previous work using learning algorithms to perform more fine-grained classification.

Wacholder et al. (1997) use hand-written rules and knowledge bases to classify proper names into broad categories. They employ an aggregation method similar to MemRun, but do not use multiple thresholds to increase accuracy.

MacDonald (1993) also uses hand-written rules for coarse named entity categorization. However, where Wacholder et al. use evidence internal to the entity name, MacDonald employs local context to aid in classification. Such hand-written heuristic rules resemble those we automatically generate.

Bechet et al. (2000) use a decision tree algorithm to classify unknown proper names into the categories: first name, last name, country, town, and organization. This is still a much coarser distinction than that focused on in this research. Further, Bechet et al. focused only on those proper names embedded in complex noun phrases (NPs), using only elements in the NP as its feature set.

7. Conclusions

The results of these experiments, though preliminary, are very promising. Our research makes clear that positive results are possible with relatively simple statistical techniques. This research has shown that training data construction is critical. The failure of our automatic data generation algorithm to produce a good sample of training data is evident in the large disparity between performances on validation and held out test sets. There are at least two reasons for the algorithm's poor sampling.

First, by using only high confidence guesses from the seed trained classifier, the training data may have a disproportionate number of instances that are easy to classify. This is evident in the number of partial names that are present in the held out test set versus the training set. Partial names, such as "Simon" instead of "Paul Simon," usually occur with weaker evidence for classification than full names. In the training set only 45.1% of the instances are partial names, whereas in the more realistic distribution of the held out set, 58.4% are partial names.

The second reason for the poor sampling stems from the use of lists of person names. Because the training set is derived from individuals in these lists, the coverage of individuals included in the training set is inherently limited. For example, in the businessperson category, lists of individuals were taken from such resources as Forbes' annual ranking of the nation's wealthiest people, under the assumption that wealthy people are often in the news. However, the list fails to mention the countless vice presidents and analysts that frequent the pages of the Wall Street Journal. This failure to include such lower level businesspersons means that a large space of the classification domain is not covered by the training set, which in turn leads to poor results on the held out test set.

The results of these experiments suggest that better fine-grained classification of named entities will require not only more sophisticated feature selection, but also a better data generation procedure. In future work, we will investigate more sophisticated bootstrapping methods, as (Collins & Singer, 1999) as well as co-training and co-testing (Muslea et al., 2000).

In future work we will also examine adapting the hierarchical decision list algorithm from (Yarowsky, 2000) to our task. Treating fine-grained classification of named entities as a word sense disambiguation problem (where categories are treated as different senses of a generic "person name") allows these methods to be directly applicable. The algorithm is particularly relevant in that it provides an intuitive way to take advantage of the similarities of certain categories (e.g., Athlete and Entertainer).

Of more theoretical concern are the problems of miscellaneous classifications that do not fit easily into any category, as well as, instances that may fit into more than one category (e.g., Ronald Reagan can be either a Politician or an Entertainer). We plan to address these issues as well as problems that may arise with extending this system for use with other classes, such as organizations.

8. References

Bechet, F., Nasr, A., Genet, F. 2000. Tagging unknown proper names using decision trees. *Proc. of ACL*, Hong Kong.

Bikel, D., Schwartz, R., Weischedel, R. 1999. An algorithm that learns what's in a name. *Machine Learning: Special Issue on NL Learning*, 34, 1-3.

Brill E. 1994. Some advances in rule based part of speech tagging. *Proc. of AAAI, Los Angeles*.

Collins, M. and Singer, Y. 1999. Unsupervised models for named entity classification. *Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Dunning, T., 1993. Accurate methods for statistics of surprise and coincidence. *Computational Linguistics*, 19:61--74.

Fellbaum, C. (ed.), 1998. *An electronic lexical database*. Cambridge, MA: MIT Press.

Fleischman, M. 2001. Automated Subcategorization of Named Entities. *Proc. of the ACL Student Workshop.*

Hastie, T., Tibshirani, R., and J. Friedman. 2001. *The Elements of Statistical Learning*. Springer.

Jurafsky, D. and Martin, J.. 2000. *Speech and Language Processing*. Upper Saddle River, NJ: Prentice Hall.

Kohavi,, R., Sommerfield, D., and Dougherty, J, 1996. Data mining using MLC++ : A machine learning library in C ++. *Tools with Artificial Intelligence*, pp. 234-245.

Lin, C.-Y. and E.H. Hovy, 2000. The Automated Acquisition of Topic Signatures for Text Summarization. *Proc. of the COLING Conference*. Strasbourg, France.

MacDonald D.D., 1993. Internal and external evidence in the identification and semantic categorization of proper names. In B.Boguraev and J. Pustejovsky, eds., *Corpus Processing for Lexical Acquisition*, pp. 61-76, Cambridge: MIT Press.

Muslea, I., Minton, S., Knoblock, C. Selective sampling with redundant views. *Proc. of the 15th National Conference on Artificial Intelligence*, AAAI-2000.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program*, 14 (no. 3), 130-137.

Quinlin, J.R., 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers.

Resnik, P. 1993. Selection and Information. PhD thesis, University of Pennsylvania.

Wacholder, N., Ravin, Y., Choi, M. 1997. Disambiguation of Proper Names in Text. *Proc. of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C.

Witten, I. & Frank, E. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with JAVA implementations*. Morgan Kaufmann, October.

Yarowsky D. 2000. Hierarchical Decision List for Word Sense Disambiguation. *Computers and the Humanities*. 34: 179-186.

Yang, Y., Pedersen, J.O., 1997. A Comparative Study on Feature Selection in Text Categorization, *Proc. of the 14th International Conference on Machine Learning ICML97*, pp. 412-420