Extracting Personal Names from Emails: Applying Named Entity Recognition to Informal Text

Einat Minkov*
Carnegie Mellon University

Richard C. Wang†
Carnegie Mellon University

William W. Cohen‡
Carnegie Mellon University

The problem of named entity recognition (NER) has been well-studied, but there has been little prior work on NER for "informal" documents—i.e., documents like email messages and bulletin board postings that are prepared quickly, and intended for a narrow audience. In this paper, we investigate NER for informal text, via an experimental study of recognizing personal names in email. We study the performance on four new annotated corpora of two machine-learning based methods: conditional random fields, and a perceptron-based method for learning HMMs. Experiments show that baseline F1 performance can be improved substantially by introduction of email-specific learning features. F1 performance is also improved by a novel recall-enhancing method, which exploits the fact that in email corpora, names are typically repeated many times across the corpus.

1 Introduction

Named entity recognition (NER), the identification of entity names in free text, is a well-studied problem. In most previous work, NER has been applied to news articles (e.g., (Bikel, Schwartz, and Weischedel, 1999; McCallum and Li, 2003)), scientific articles (e.g., (Craven and Kumlien, 1999; McCallum et al., 2000; Bunescu and Mooney, 2004)), or web pages (e.g., (Freitag, 1998; Etzioni et al., 2004)). These genres of text share two important properties: documents are written for a fairly broad audience, and writers take some care in preparing documents. Important genres that do *not* share these properties include instant messaging logs, email messages, and (to a somewhat lesser extent) call center logs, bulletin board postings, and newsgroup postings. Below we will call text from these genres "informal" text.

NER from informal text has many possible uses: for email alone, potential applications include improved search over large collections of email, email prioritization, social network analysis, and semi-automatic meeting negotiation and scheduling. However, informal text is harder to process automatically. Since less time is spent in preparing and reviewing informal documents, they contain more grammatical and spelling errors; further, since the audience is more restricted, they contain more abbreviations, and more group- and task-specific jargon. Because of these differences, existing NER methods may

* Language Technology Institute, Pittsburgh, PA 15213, US. E-mail: einat@cs.cmu.edu
† Language Technology Institute, Pittsburgh, PA 15213, US. E-mail: rcwang@cs.cmu.edu
‡ Center for Automated Learning and Discovery, Pittsburgh, PA 15213, US. E-mail: wcohen@cs.cmu.edu

require modifications to perform well on informal text.

In this paper, we investigate NER for informal text with an experimental study of the problem of recognizing personal names in email—a NER task that is both useful and non-trivial. There are several contributions of this work.

First, we present four corpora of informal, email-like text for personal names, each roughly comparable in size to the MUC-6 corpus. Three of these are publicly available (the fourth cannot be distributed due to privacy considerations).

Second, we experimentally evaluate two existing state-of-the art machine-learning based NER methods—specifically, conditional random fields (CRFs) (Lafferty, McCallum, and Pereira, 2001), and a perceptron-based method for learning hidden Markov models (HMMs) (Collins, 2002a). These methods are supported by different types of formal analysis: CRFs are a probabilistic approach to labeling sequences of examples, and perceptron-trained HMMs are a margin-based approach. We show that these methods achieve comparable performance on this problem.

Third, we evaluate several techniques for improving the performance of NER methods for this specific task. The simplest-to-implement of these techniques are new word features that exploit properties of email; for instance, words that appear at the end of a message are more likely to be names. Experiments show that these email-specific features significantly improve NER performance on our corpora.

We also present and evaluate novel techniques for exploiting *repetition* of names in a test corpus. Email messages often include some structured, easy-to-recognize names, such as names within a header, or names appearing in automatically-generated phrases (like “On August 16, 2004, Richard C. Wang writes:”, which might precede a quoted message). However, other names occur in highly ambiguous contexts. As an example, a message body might contain only one sentence, “can we get rich?”. Depending on context, this might or might not mean “Can we get Richard to help us?”

This observation (supported by some analysis that we present below) suggests that techniques for exploiting discourse are important in obtaining high recall NER. Techniques for exploiting name repetition within documents have been previously applied to newswire text (e.g., (Humphreys et al., 1998)) and scientific abstracts (e.g., (Bunescu et al., 2002)); however, we believe that discourse in email is substantially different, and hence requires different techniques.

In formal documents, discourse begins by establishing a common context with the reader, and the discourse is limited to a *single document*; hence a news story often begins by giving a longer, clearer version of an entity’s name (e.g., “The Walt Disney Corporation”) which is later followed by an abbreviated, more ambiguous version (e.g., “Disney”). However, in email, context is not always established within a single documents. Instead, the sender and receiver typically share a context, which must be inferred. In the method we present, this context is inferred by examining *multiple documents* in a corpus.

To exploit names that recur across multiple documents, we present a simple and efficient scheme for increasing recall. In brief, the method considers re-assigning labels to words w which sometimes occur inside a predicted NE, and sometimes outside of any predicted NE. Re-assignment decisions are based on a “weight” for w, which can be interpreted as a probabilistic determination of whether the “one sense per discourse” assumption (Yarowsky, 1995) holds for w. Experiments show that this technique always improves recall substantially, and almost always improves F1 performance.

A final issue considered in this paper is the question of what our NER systems have learned—in particular, to what extent have they learned to recognize the *context* in which names appear, and to what extent have they learned the *content* of entity names (i.e., the particular names that appear in a corpus). This issue is of importance because often, a small number of distinct names account for most of the name occurrences in email drawn from a particular working group. However, one would like a NER for that group's email to maintain the same level of performance, even if the people in the group change over time. In other words, one would prefer the NER to rely primarily on context properties, rather than content. This issue will be considered at several points throughout the paper.

In the remainder of the paper, we will first describe the corpora we used. Next, in Section 3, we will illustrate the applicability of state-of-the-art sequential learners for extracting personal names from emails. Section 4 presents an analysis of how names repeat in email, and gives some upper bounds on the potential for exploiting name repetition to enhance recall. Section 5 describes our approach for enhancing recall by exploiting name repetition across multiple documents. The paper is concluded with a discussion of related work and a summary of our contributions.

2 Corpora

Due to privacy issues, large and realistic email corpora are rarely available for research purposes: for instance, in most US academic institutions, a users' email can only be distributed to researchers if *all senders* of the email provide explicit written consent. Thus obtaining email for studies such as this one is a non-trivial problem.

The first email corpora used in our experiments was extracted from the CSpace email corpus (Kraut et al., 2004), which contains approximately 15,000 email messages collected from a management course conducted at Carnegie Mellon University in 1997. In this course, 277 MBA students, organized in approximately 50 teams of four to six members, ran simulated companies in different market scenarios over a 14-week period. All email of the students associated with this course was collected. *Mgmt-Game* is a subcorpora consisting of all emails written over a five-day period. In the experiments, four days worth of email were used as a training set, and the fifth day as a test set.

This collection of mail is to some extent artificial, as the people who wrote it were involved in a "management game", in which the goal was to simulate companies. In another sense, however, the corpus is quite real: the students who wrote it were enrolled in a real course, with real assignments and real grades. Many of the emails in this corpus involve negotiation and delegation of meetings and shared tasks (Cohen, Mitchell, and Carvalho, 2004). We believe the corpus to be quite similar to the work-oriented mail of employees of a small or medium-sized company.

The email corpus saved in the management-game experiment had been preserved in a relational database in a preprocessed form. In particular, message headers had been parsed into a set of pre-defined fields. While convenient for many purposes, the preprocessing unfortunately eliminated whatever variation in header formats might have been present in the original email. To reduce the bias of training on the highly regular headers that we regenerated from the database, the text corpus we constructed contains only three header fields: "From", "Subject", and "Time". (This was considered the minimal context needed for correct labeling.)

The next two collections of email were extracted from the Enron corpus (Klimt and Yang, 2004). This data was originally made public and posted to the web by the Federal Energy Regulatory Commission during its investigation. A version of this data was later purchased by the CALO project (CALO, 2004) and subsequently made available for research purposes. We formed two subsets of this data. The first subset, *Enron-meetings*, consists of all messages in folders named "meetings" or "calendar", with two exceptions: (a) six very large files were removed, and (b) one very large "calendar" folder was excluded. Most but not all of these messages were meeting-related. The second subset, *Enron-random*, was formed by repeatedly sampling a user name (uniformly at random among the 158 users), and then sampling an email from that user (uniformly at random).

As a second type of informal text, we also annotated a collection of newsgroup postings, taken from the 20Newsgroups corpus (Craven et al., 2000). This collection is a subset of the corpora known to contain complex "signatures", a common construct which seems to be under-represented in the CSpace and Enron corpora. It was collected by Vitor Carvalho as part of a related project on learning to extract email signatures (Carvalho and Cohen, 2004).

Annotation was done by presenting annotators with messages marked up with a rule-based NER system as a baseline. Annotators were instructed to include nicknames and misspelled names, but exclude person names that are part of an email address (e.g., "william.cohen@cs.cmu.edu") and names that are part of a larger entity name like an organization or location (e.g., "David A. Tepper School of Business").

The sizes of the various corpora are summarized in Table 1. The table refers to additional two datasets, namely MUC6 and Mgmt-Game-Groups, which will be described below.

Corpus	# Documents		#Tokens	#Names	#Names per Email
	Train	Test			
Mgmt Game	749	264	139,865	2,993	3.0
Enron-meetings	729	247	204,423	2,868	3.0
Enron-random	516	164	285,652	5,059	7.4
NewsGroups	477	120	300,177	2,885	4.8
MUC-6	347	30	204,071	2,559	6.8
Mgmt-Game-groups	631	128	104,662	2,792	3.7

Table 1
Summary of the corpora used in the experiments. (For MUC-6, our training set combines the "training" and "dry run" subsets.)

3 Experiments with Existing NER Methods

A common approach to NER is to reduce it to the task of *tagging* (i.e., classifying) each word in a document. Specifically, a document is encoded as a sequence $\mathbf{x}$ of tokens $x_1, \ldots, x_N$, and a *tagger* associates with $\mathbf{x}$ a parallel sequence of tags $\mathbf{y} = y_1, \ldots, y_N$, where each y_i is in some tag set Y. If these tags are appropriately defined, the entity names can be derived from them. Here we consider two tag sets. The *binary* tag set has two labels y—one label for tokens inside an entity, and one label for tokens outside an entity. The *multi* tag set has five tags y, corresponding to (1) a one-token entity, (2) the first token of a multi-token entity, (3) the last word of a multi-token entity, (4) any other token of a multi-token entity and (5) a token that is not part of an entity.

A common way of constructing such a tagging system is to learn a mapping from $\mathbf{x}$ to $\mathbf{y}$ from labeled data (e.g., (Bikel, Schwartz, and Weischedel, 1999; Borthwick et al., 1998; Malouf, 2002)). Annotated documents are converted to $(\mathbf{x}, \mathbf{y})$ pairs. In addition to specifying a tagging scheme, it is necessary to specify a *feature extractor*, i.e., a function that converts an occurrence of a token w inside a document into a set of descriptive features.

In our first set of experiments we apply two state-of-the-art machine learning methods, which have been previously used for NER, and evaluate them on the problem of extracting personal names from email. The methods have been described in detail elsewhere, but we will summarize them here for completeness. We will then describe in detail the features used in our NER experiments.

3.1 VP-HMM: A margin-based sequential learner

Many methods for learning taggers exploit, in some way, the sequential nature of the classification process. In general, each tag depends on the tags around it: for instance, if person names are usually two tokens long, then if y_i is tagged "Person" the probability that y_{i+1} is "Person" is increased, and the probability that y_{i+2} is "Person" is decreased. Hence many learning-based approaches to NER learn a sequential model of the data, generally some variant of a hidden Markov model (HMM)(Durban et al., 1998).

We experimented with two sequential learning methods: Collins' voted-perceptron based algorithm for discriminatively training HMMs (Collins, 2002a) (below, VP-HMM), and conditional random fields (CRFs) (Lafferty, McCallum, and Pereira, 2001).

The VP-HMM method can be summarized as follows. Assume a *local feature function* $\mathbf{f}$ which maps a pair $(\mathbf{x}, \mathbf{y})$ and an index i to a vector of features $\mathbf{f}(i, \mathbf{x}, \mathbf{y})$. Define

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) = \sum_{i}^{|\mathbf{x}|} \mathbf{f}(i, \mathbf{x}, \mathbf{y})$$

and let W be a weight vector over the components of $\mathbf{F}$. Also let $V(W, \mathbf{x})$ denote the Viterbi decoding of $\mathbf{x}$ with W, *i.e.*,

$$V(W, \mathbf{x}) = argmax_{\mathbf{y}} \mathbf{F}(\mathbf{x}, \mathbf{y}) \cdot W$$

To make Viterbi search tractable, we must restrict $\mathbf{f}(i, \mathbf{x}, \mathbf{y})$ to make limited use of $\mathbf{y}$. Here we assume that for each component f^k of $\mathbf{f}$,

$$f^k(i, \mathbf{x}, \mathbf{y}) = f^k(g^k(i, \mathbf{x}), y_i, y_{i-1})$$

where $g^k(i, \mathbf{x})$ computes some property of the i-th component of $\mathbf{x}$.

The goal of learning is to find a W that leads to the globally best overall performance of a tagger based on Viterbi decoding. This "best" W is found by repeatedly updating W to improve the quality of the Viterbi decoding on a selected example $(\mathbf{x}_t, \mathbf{y}_t)$.

Specifically, Collin's algorithm starts with $W_0 = \mathbf{0}$. After the t-th example $\mathbf{x}_t, \mathbf{y}_t$, the Viterbi sequence $\hat{\mathbf{y}}_t = V(W_t, \mathbf{x}_t)$ is computed, and W_t is replaced with

$$W_{t+1} \quad = \quad W_t + \mathbf{F}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{F}(\mathbf{x}_t, \hat{\mathbf{y}}_t) \qquad (1)$$

Examples are presented in multiple *epochs*. In each epoch all the documents are randomly re-ordered, and then each is presented as a $\mathbf{x}_t, \mathbf{y}_t$ pair. The number of epochs E is a parameter of the algorithm. After training, one takes as the final learned weight vector W the average value of W_t over all time steps t.

This simple algorithm has performed well on a number of important sequential learning tasks (Collins, 2002a; Altun, Tsochantaridis, and Hofmann, 2003; Sha and Pereira, 2003), including NER. It can also be formally proved to converge under certain plausible assumptions (Collins, 2002a).

3.2 CRF: A probabilistic sequential learner

A CRF is a probabilistic conditional model $\Pr(\mathbf{y}|\mathbf{x})$ which defines a probability distribution over tag vectors $\mathbf{y}$ given word vectors $\mathbf{x}$. This model is based on a Markov random field, with nodes corresponding to elements of the structured object $\mathbf{y}$, and with potential functions that are conditional on (features of) $\mathbf{x}$. Learning is performed by setting parameters to maximize the likelihood of a set of $(\mathbf{x}, \mathbf{y})$ pairs given as training data. For NER problems, the Markov field is a chain, and $\mathbf{y}$ is a linear sequence of word tags.

As before, we assume a set of features $\mathbf{f}$. Following the notation of Sha and Pereira (Sha and Pereira, 2003) we define a conditional random field (CRF) to be an estimator of the form

$$\Pr(\mathbf{y}|\mathbf{x}, W) = \frac{1}{Z(\mathbf{x})} e^{W \cdot \mathbf{F}(\mathbf{x},\mathbf{y})} \tag{2}$$

where W is again a weight vector over the components of $\mathbf{F}$, and $Z(\mathbf{x}) = \sum_{\mathbf{y}'} e^{W \cdot \mathbf{F}(\mathbf{x},\mathbf{y}')}$. Although the definition of $Z(\mathbf{x})$ involves a summation over exponentially many label vectors $\mathbf{y}'$, it can be computed efficiently using a dynamic programming scheme, similar to the forward-backward algorithm.

In training the goal is to maximize log-likelihood over the training set. The CRF implementation we use performs this maximization with a limited-memory quasi-Newton method (Liu and Nocedal, 1989; Malouf, 2002). We also use a Gaussian prior to reduce overfitting, which amounts to introducing a penalty for the L_2 norm of the weight vector W.

Like the epochs of VP-HMM, the optimization method makes multiple passes over the data. In each pass certain expectations are computed with respect to the current weight vector, which requires running the forward-backward algorithm for each example. In our experiments we observed that in spite of the Gaussian prior, the optimization does sometimes overfit the training data; also, the optimization can take a very long time to converge. To control overfitting and reduce runtime, we introduced as an additional parameter a bound on the number of iterations of the optimization method, similar to the number of epochs E for the VP-HMM.

The implementations of CRF and VP-HMM that we used for these experiments are from the Minorthird package (Minorthird, 2004).

3.3 Features

We trained and tested the learners using two sets of features, referred to below as the *basic* and *extended* feature sets. In the notation used above, these features are $g(i, \mathbf{x})$, where $\mathbf{x}$ is an array of tokens and i indexes a token t in the array; below t will be called a *focus word*. Notice that $g(i, \mathbf{x})$ must be a real-valued function, while the natural features we would like to define have string values; for instance, one natural feature $g(i, \mathbf{x})$ might simply return the focus word. We handle this mismatch in the usual way, by introducing binary indicator functions g that have the "natural value" embedded in their names. For instance, when we observe an $\mathbf{x}$ where the focus word is "william", we might introduce a new function $g_{\text{focusWord.william}}(i, \mathbf{x})$ that has value 1 iff $x_i =$"William". Below we will discuss the functions as if they returned string values. Table 2 gives the internal notation used as well as the value of the "natural" string-valued function.

The *basic features* for a token t include the lower-cased value of t, and the *capitalization pattern* for t. The capitalization pattern is constructed by replacing all capital letters with the letter "X", all lower-case letters with "x", and all digits with "9", and the compressing runs of the same letter with a single letter. All features are also computed for a window of 3 tokens to the left and to the right of the focus word t. Some examples are given in Table 2.

The *extended features* exploit some additional relevant information, as detailed below.

- *Dictionary features:* We use several name dictionaries as features. One is the US Census' lists of the most common first and last names in the US, reflecting year 1990 records, and including 5,492 and 88,798 names respectively.[1] Another

1 These were taken from http://www.census.gov/genealogy/www/freqnames.html.

Feature function $g(i, \mathbf{x})$	Feature name and value
Basic Features	
focus word, in lowercase	lc.cohen=1
one token to left of focus word, lowercased	left.1.lc.william=1
two tokens to left ...	left.2.lc.is=1
three tokens to left ...	–
one token to right of focus word	right.1.lc.attending=1
two tokens to right	right.2.lc.nips=1
three tokens to right	right.3.lc.this=1
capitalization of focus word	cap.Xx=1
capitalization one token to left	left.1.cap.x=1
...	
Extended Features	
indicates if the focus word is in the “from” field	inFrom=1
indicates if token to left of focus is in the “from” field	left.1.inFrom=1
...	
indicates if the focus word is in a “signoff” (≈ after two line breaks and near end of message)	inSignoff=1
indicates if token to left of focus is in “signoff”	left.1.inSignoff=1
...	
indicates if the focus word is a probable initial (X or X.)	isInitial=1
indicates if the token to the left of focus is a probable initial	left.1.isInitial=1
...	
indicates if the focus word is part of an email address (regex)	isEmail=1
indicates if the focus word appears anywhere in the header	inHeader=1
indicates if the focus word starts a new sentence (≈ capitalized after a period, question or exclamation mark)	startSentence=1
indicates if the focus word is in the common words dictionary	isCommon=1
indicates if the focus word is in the first names dictionary	FirstName=1
indicates if the focus word in the last names dictionary	LastName=1
indicates if the focus word in roster names dictionary	inRoster=1
indicates if the focus word is considered a “sure first name” (FirstName AND not LastName AND not isCommon)	SureFirst=1
indicates if the focus word is considered a sure last name	SureLast=1
indicates if the focus word is considered a sure name ((SureFirst OR SureLast OR inRoster) and not isCommon)	SureName=1
indicates if the focus word is in the personal prefixes/suffixes dict.	PersonalTitle=1
indicates if the focus word is in the organizations suffixes dict.	OrgSuffix=1
indicates if the focus word is in the location suffixes dict.	LocSuffix=1
indicates if the focus word is followed by the bigram ”and I”	beforeAndI=1
indicates if the focus word is referred to by a pronoun (≈ capitalized and followed by a sing. pronoun within 15 tokens)	hasPronoun=1
...	

Table 2
Examples of the various features used in the experiments. The “basic” features are applied at position $i = 3$ to the sample document “Is William Cohen attending Nips this fall? - Richard”. The “extended” features are applied to hypothetical tokens on which they give a “true” value.

is a dictionary of 16,623 student names, obtained from students at universities across the country as part of the RosterFinder project (Sweeney, 2003). We also used a dictionary of the base forms, conjugations and plural forms of common English words, and a relatively small ad-hoc dictionary representing words especially common in emails (e.g., “email”, “inbox”). Finally we used some small manually created word dictionaries of prefixes and suffixes indicative of persons (e.g., "mr", "jr"), locations (e.g., "ave") and organizations (e.g., "inc").

The dictionaries were used to define various categories of words including common words, first names, last names and "roster names". (A roster name is any word from the roster-name dictionary, in which first and last names are mixed.) In addition, we constructed three composite features that combine membership in a name dictionary with non-membership in the common-word dictionary. A word that is in the first names dictionary and is not in the common words dictionaries is designated a "sure first name". The feature "sure last name" is defined analogously, and words that are in any of the name dictionaries but not the common-word dictionary are designated "sure names".

- *Email structure features:* We perform a simplified document analysis of the email and use this to construct some additional features. One is an indicator as to whether a token t is contained in the "from" field of an email. Another indicates whether a token in the email body is equal to some token t appearing in the header. An indicator feature based on a regular expression is used to mark tokens that are part of a probable "sign-off" (i.e., a name at the end of a message). Finally, since the annotation rules do not consider email addresses to be names, we added an indicator feature for tokens that are inside an email address.

- *Syntactic features*: We experimented, unsuccessfully, with using features derived from POS tags and NP-chunking of the email. We conjecture that the POS tags assigned by our tagger (a version of Brill's WSJ-trained tagger (Brill, 1995)) are too noisy to be beneficial. We did include some features based on linguistic rules. One rule looks for capitalized words that are not common words and are followed by a pronoun within a distance of up to 15 tokens. (As an example, in the document "Try to contact Puck tomorrow. He should be around." a human reader will realize that Puck is a person, rather than, say, an organization). Another rule looks for words followed by the bigram "and I". As is common for hand-coded NER rules, both these rules have high precision and low recall.

3.4 Experiments

3.4.1 Feature sets and parameter settings

In our initial experiments, the Mgmt-Game dataset was used as a development set to refine and evaluate the features and parameter settings. We begin by analyzing the effect of the number of epochs E on entity-level F1 measure, for a combination of feature sets and tag sets. (We note that our use of VP-HMM differs somewhat from previous experiments, in which the number of epochs was generally smaller (Collins, 2002a).) Table 3 shows the results of these experiments. In the table, the F1 measure that is best, over all epochs, for a particular algorithm, tag-set, and feature-set is in bold.

To evaluate these results further, we split each test set into five disjoint subsets, and computed the F1 measure on each subsets. These five measurements allow one to compute a standard error for each F1 measurement (shown in parentheses in the table). The five measurements also allow two different F1 measures to be statistically compared.

From the statistical tests, it is clear that the extended features do indeed improve performance over the basic features (for all of the sixteen cases, a paired t-test gives confidence level of $p > 0.975$). It is also evident that the multi-tags are preferable to the binary tags in this case: with extended features, the best results for multi-tags are significantly better ($p > 0.95$) than the best results with binary tags.

Algorithm	Epochs	Binary Tags		Multi-tags	
		Basic	Extended	Basic	Extended
VP-HMM	5	84.3 (0.7)	92.3 (0.3)	88.2 (0.8)	94.7 (0.4)
	20	90.6 (0.3)	93.6 (0.4)	91.6 (0.6)	94.5 (0.8)
	50	91.0 (0.5)	94.2 (0.6)	**92.0** (0.2)	94.8 (0.9)
	100	**91.9** (0.4)	**94.5** (0.5)	91.8 (0.3)	**95.9** (0.3)
CRF	5	47.4 (1.3)	72.9 (1.2)	43.8 (1.0)	80.8 (1.2)
	20	83.4 (0.5)	92.9 (0.5)	86.2 (1.0)	94.5 (0.5)
	50	90.4 (0.7)	**94.3** (0.8)	91.2 (0.5)	**95.5** (0.3)
	100	**91.3** (0.4)	94.2 (0.9)	**91.8** (0.6)	95.3 (0.4)

Table 3
Entity-level F1 measurements and standard error (in parentheses) for the Mgmt game corpus, varying the number of epochs, features and tag-sets.

The experiments suggest that the number of epochs depends on several factors. Overall, the VP-HMM converges more quickly than the CRF, and hence it may be useful when training time is critical. Also, the extended feature sets require fewer epochs to train. For the extended feature sets, performance stabilizes after around 50 epochs—the differences between 50 and 100 epochs are generally not statistically significant[2]. Since computation time for the experiments is a consideration, subsequent results given in the paper will be for 50 epochs only. We will also consider only the multi-tag tag set, unless otherwise noted.

Finally, the experiments suggest that the two algorithms perform comparably well, when given sufficient time for learning. The best results of VP-HMM and CRF (the ones boldfaced in the table) are not statistically significantly different for any combination of feature and tag set.

3.4.2 Context *vs* content with basic and extended features We now turn to the question of what our NER systems have learned—in particular, to what extent have they learned to recognize the linguistic *context* in which names appear, and to what extent have they memorized the particular entity names associated with the Mgmt-Game corpus. Below we will take a rather broad notion of "context", and include as "context" not only the words surrounding a token, but the relationship of that token to other external objects—in particular, the membership of that token in external dictionaries.

As one way of measuring the relative contribution of context *vs.* content, we removed the feature for the focus word from the basic feature set. This makes it more difficult for the learner to memorize names from the training set, and simulates a situation where the test set includes names that were not encountered before. The result of this experiment on the Mgmt-Game dataset is shown in Table 4(a). As might be expected, removing the focus-word features degrades performance. However, performance degrades much more when the basic features are used than when the extended features are used. This suggests that extractors learned using the extended feature set rely more on context and less on content than those learned using basic feature set.

As confirmation of the generality of the effect, a similar experiment conducted on the Enron-meetings dataset. The results are similar, as is shown in Table 4(c).

In an additional experiment, we constructed another split of the Mgmt-Game corpus to evaluate the importance of content information. In the management game task, users were divided into 50 teams, and most email was exchanged between members of a team.

2 The only exception to this is the VP-HMM with binary tags and basic features.

Algorithm	Basic Features			Extended Features		
	focus	no focus	Δ	focus	no focus	Δ
VP-HMM	92.0	77.1	16.2%	94.8	94.1	0.7%
CRF	91.2	77.3	15.2%	95.5	93.1	2.5%

(a) Mgmt-Game corpus.

Algorithm	Basic Features			Extended Features		
	focus	no focus	Δ	focus	no focus	Δ
VP-HMM	76.4	70.2	8.1%	91.2	88.6	2.9%
CRF	75.0	71.0	5.3%	88.4	86.8	1.8%

(b) Mgmt-Game-Teams corpus.

Algorithm	Basic Features			Extended Features		
	focus	no focus	Δ	focus	no focus	Δ
VP-HMM	75.9	66.3	12.6%	79.8	77.3	3.2%
CRF	73.0	68.6	6.0%	79.7	78.5	1.5%

(c) Enron-Meetings corpus.

Table 4
Entity F1, with and without "focus word" features. In Mgmt-Game and Enron-Meetings, performance is less sensitive to the presence of the focus-word feature when the extended feature set is used. The same behavior is seen in Mgmt-Game-Teams. Comparing Mgmt-Game to the more difficult Mgmt-Game-Teams task, performance degrades dramatically when the basic features are used, but only slightly when the extended feature set is used.

We split the corpus into a training and test set such that the senders of training-set messages were on different teams from senders of test-set messages—i.e., messages from a single team were never split between the training and test sets. Therefore, an extractor is trained on messages from some working groups, and then tested on messages from other groups. This is a more rigorous simulation of testing on a different distribution of person names from that seen in the training set. Since this task is clearly harder than training and testing on a more homogenous corpus, we should expect somewhat worse performance.

Table 4(b) shows the results of this experiment. Comparing these results to Table 4(a) shows that performance is indeed lower than on Mgmt-Game. However, for this split, the F1 performance for the extended feature set is far better than for the basic feature set—the best extended-feature F1 is 91.2%, while the best basic-feature F1 is only 76.4%. This again supports the conjecture that the extended features emphasize content versus context in the learned classifier. Also, as before, the performance of the basic feature set is much more sensitive to the focus word feature.

To summarize the results of the last two sections, there appear to be no large differences between the two learners VP-HMM and CRFs, when comparable training time is provided for each. However, the multi-tag tag scheme is better than the binary scheme, and the extended feature set is better than the basic feature set. The extended feature set is also more robust to changes to the distribution of names in the test set.

3.4.3 Performance on different datasets Table 5 summarizes performance of the extended-features and basic-features for each learner on several different corpora of informal text: Mgmt-Game, Mgmt-Game-Teams, Enron-Random, Enron-Meetings, and NewsGroups. In addition to entity-level F1, denoted as E-F1, we also show token-level

Fea. Set	Dataset	VP-HMM				CRF			
		T-P	T-R	T-F1	E-F1	T-P	T-R	T-F1	E-F1
Extended	Mgmt Game	97.7	96.4	97.1	94.8	98.2	95.9	97.1	**95.5**
	Mgmt Teams	97.9	90.2	93.9	***91.2**	98.2	89.9	93.9	88.4
	Enron-Random	92.1	90.5	91.3	***86.1**	90.9	88.3	89.6	83.8
	Enron-Meetings	92.2	87.1	89.6	**79.8**	93.5	84.3	88.7	79.7
	NewsGroups	94.2	74.4	83.1	70.5	91.6	77.4	83.9	**70.9**
Basic	Mgmt Game	98.4	92.7	95.5	**92.0**	97.6	92.2	94.8	91.2
	Mgmt Teams	96.5	76.8	85.5	**76.4**	97.3	74.2	84.2	75.0
	Enron-Random	93.5	80.2	86.4	**81.6**	91.3	82.6	86.7	79.8
	Enron-Meetings	91.5	75.7	82.8	***75.9**	89.3	77.4	82.9	73.0
	NewsGroups	91.6	63.0	74.7	60.6	89.7	65.7	75.8	**62.2**

Table 5
Performance for the two sets of features and multi-tags, across all datasets.

precision, recall and F1, denoted as T-P, T-R, and T-F1 respectively.[3] In the table, the best entity level F1 results among the two learners for each dataset and feature set are marked in bold. An asterisk designates an F1 measure that is significantly better than the result for the lower-scoring learner.

Overall the level of performance is encouraging. In the four email-related datasets, entity-level F1 performance is reasonably good. Performance on Mgmt-Game-Teams is lower than for Mgmt-Game mainly because (by design) there is less similarity between training and test sets with this split. Enron emails seem to be harder than Mgmt-Game emails, perhaps because they include relatively fewer structured instances of names. Enron-Meetings emails also contain a number of constructs that were not encountered in the Mgmt-Game corpus, notably lists (e.g., of people attending a meeting), and also include many location and organization names, which are rare in Mgmt-Game. A larger set of dictionaries might improve performance for the Enron corpora.

Not unexpectedly, performance is worst on NewsGroups, the only non-email corpus of informal text we experimented with. The NewsGroups dataset was taken from several newsgroups from different domains, and contains a broad mixture of multiple styles. In addition to email-like messages, it contains messages quoted from (or similar to) newswire, literature or even the Bible. As noted above, these messages were also extracted because they contain non-trivial "signatures"—which of course contain names in linguistically unusual contexts. We believe that the email-specific features are not optimally suited for this genre of informal text, and that better performance would be obtained if the genre were modeled separately.

3.4.4 Email vs Newswire For comparison, we applied the same learning methods to the MUC-6 corpus. Table 6 shows results for VP-HMM and CRF with both the basic and extended feature sets. Performance on MUC-6 is generally not competitive with the best previously-reported results on this task. For MUC-6, the extended features do not statistically significantly improve performance for either learner—in fact for CRFs, performance with the basic feature set is significantly better than with the extended feature set. This is unsurprising, given that the extended emails were designed for email, and supports the hypothesis that NER is somewhat different for informal text in general, and email in specific.

3 Token-level recall is the ratio of the number of tokens that were marked by the extractor as inside a name to the total number of tokens truly inside a name. Token-level precision and F1 are defined analogously.

Algorithm	Basic Features	Extended Features
VP-HMM	87.6	**89.7**
CRF	***88.7**	81.6

Table 6
Entity F1 results for the MUC-6 dataset with different feature sets, multi-tags. On this dataset, the email-specific extended features do not statistically significantly improve performance for either learner. For CRFs, the extended features significantly degrade performance.

left.2.mr	left.1.president
left.2.mrs	left.2.dr
left.1.jr	right.2.who
left.1.judge	right.2.jr
right.3.staff	left.3.by
left.2.ms	right.3.president
right.2.staff	left.3.by
right.1.family	left.3.rep
left.3.says	left.2.rep
right.3.resporter	right.1.administration

left.1.by	right.2.home
left.2.by	right.1.or
left.3.name	left.1.with
left.2.name	left.1.thanks
left.3.by	right.1.picked
right.3.his	left.3.meet
right.1.ps	right.1.started
right.3.home	right.1.told
right.1.and	left.2.prof
left.1.called	left.2.email

Figure 1
Predictive contexts for personal-name words for MUC-6 (left) and Mgmt-Game (right) corpora.

To further explore the differences between email and newswire NER problems, we stripped all header fields from the Mgmt-Game messages, and trained a model (using VP-HMM and basic features) from the resulting corpus of email "bodies". We then compared the features used in this model with the features used in the corresponding model learned from the MUC6 corpus. Figure 1 shows the features most indicative of a token being part of a name in these two models. To make the list easier to interpret, it includes only the features corresponding to tokens surrounding the focus word.

As one might expect, the important features from the MUC-6 dataset are mainly formal name titles such as "mr", "mrs", and "jr", as well as job titles and other pronominal modifiers such as "president" and "judge". However, for the Mgmt-Game corpus, most of the important features are related to email-specific structure. For example, the features "left.1.by" and "left.2.by" are often associated with an quoted excerpt from another email message, which in the Mgmt-Game corpus is often marked by mailers with text like "Excerpts from mail: 7-Sep-97 Re: paper deadline by Einat Minkov". Similarly, features like "left.1.thanks" and "right.1.ps" indicate a "signoff" section of an email, as does "right.2.home" (which often indicates proximity to a home phone number appearing in a signature).

4 Repetition of named entities in email

In the results of Table 5, the extractors uniformly have high precision, but relatively low recall. This suggests that some sort of recall-enhancing procedure should be adopted to improve overall performance. One family of recall-enhancing techniques are based on looking for multiple occurrences of names in a corpus, so that names which occur in ambiguous contexts (e.g., "can we get rich?") will be more likely to be recognized.

In the introduction, we conjectured that the ways in which names repeat themselves in a corpus would be different in email and newswire text. In news stories, one would expect repetitions within a *single document* to be common, as a means for an author to

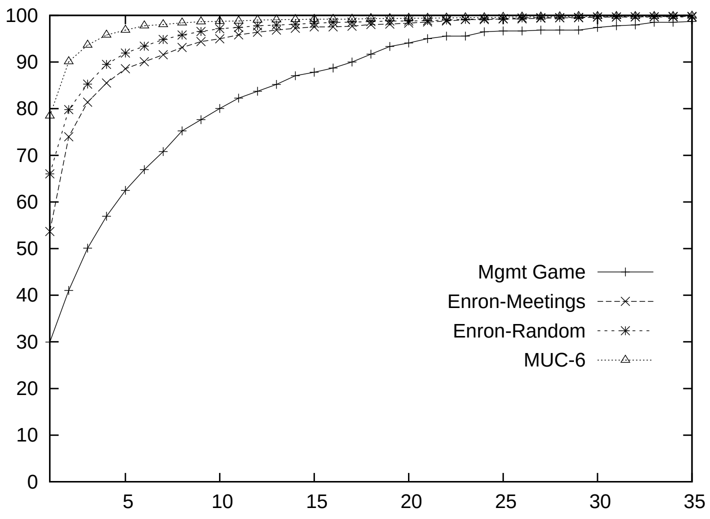


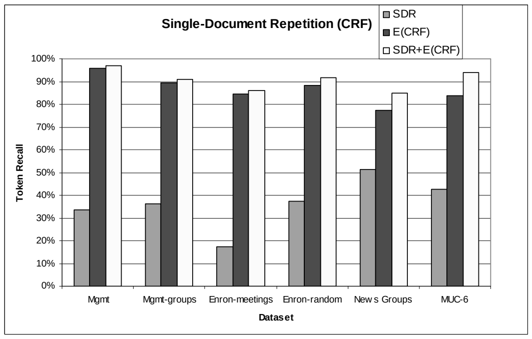
Figure 2
Cumulative percentage of person-name tokens w that appear in at most K distinct documents as a function of K.

establish a shared context with the reader. In an email corpus, one would expect names to repeat more frequently across the corpus, in *multiple documents*—at least when the email corpus is associated with a group that works together closely.

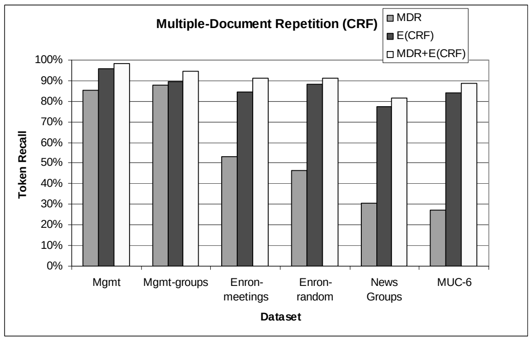
In the remainder of this section, we will present some analysis that supports this conjecture. Specifically, we will show that for our email corpora, repetitions of names across multiple documents are more common than for newswire data, and hence that recall-enhancing methods should be based on looking at terms that occur across documents. We will also show that only a modest improvement can be expected on our email corpora by exploiting only single-document repetitions.

In a first experiment, we plotted the percentage of person-name tokens w that appear in at most K distinct documents as a function of K. Figure 2 shows this function for the Mgmt-Game, MUC-6, Enron-Meetings, and Enron-Random datasets. There is a large separation between MUC-6 and Mgmt-Game, the most workgroup-oriented email corpus. In MUC-6, for instance, almost 80% of the names appear in only a single document, while in Mgmt-Game, only 30% of the names appear in only a single document. At the other extreme, in MUC-6, only 1.3% of the names appear in 10 or more documents, while in Mgmt-Game, almost 20% do. On average, a name appears in 1.7 documents in the MUC-6 corpus, and in 6.9 documents in the Mgmt-Game corpus. The Enron-Random and Enron-Meetings datasets show distributions of names that are intermediate between Mgmt-Game and MUC-6. The name distribution on the NewGroups dataset (omitted from the graph) is nearly identical to that of MUC-6.

As a second experiment, we implemented two very simple extraction rules. The *single document repetition* (SDR) rule marks every token that occurs more than once inside a single document as a name: for instance, both instances of the token "puck" would be marked in the following document (where italicized words are names): "Ask *dr. puck* if we can have an extension . . . if we have no luck with *puck* then we'll have to download an old . . . " Adding tokens marked by the SDR rule to the tokens marked by an extractor E generates a new extractor, which we will denote SDR+E. Clearly, SDR+E has a higher token recall than E, and further, it is clear that the recall of SDR+E is an upper bound on the token recall of any recall-enhancing method that improves E by exploiting repetition



(a) SDR



(b) MDR

Figure 3
Upper bounds on recall and recall improvements associated with methods that look for terms that re-occur within a single document (SDR) or across multiple documents (MDR).

within a single document.

Analogously, the *multiple document repetition* (MDR) rule marks every token that occurs in more than one document as a name. Again, the token recall of MDR+E rule is an upper bound on the token recall of any recall-enhancing method that exploits token repetition across multiple documents. (Note that token repetitions repetitions within a single documents are *not* marked: for instance, “puck” would not be marked in the example above. This is a somewhat artificial restriction, which is not imposed by the recall-enhancement algorithm we will describe in Section 5; however, it is helpful to make this distinction in the analysis that follows.)

The light gray bars in Figure 3 show the recall obtained by the SDR rule (on the left) and the MDR rule (on the right). The MDR rule has highest recall for the two Mgmt-Game corpora, and lowest recall for the MUC-6 corpus and the NewsGroups corpus. Conversely, for the SDR rule, the two highest recall levels obtained by the SDR rule are for NewsGroups and MUC-6.

Figure 3 also shows the token recall obtained by the CRF extractor (using extended fetaures and multi-tags), in dark gray. In white, the figure shows the token recall of the SDR+E and MDR+E extractors, where E stands for the CRF extractor. The white bars in the left-hand chart are thus an upper bound on recall for SDR-like methods—methods which which exploit single-document repetition. Comparing them to the dark gray bars, we see that the maximal potential recall gains from a SDR-like method are small on the email corpora, but larger on MUC-6 and NewsGroups. For MDR-like methods, there are large potential gains on Mgmt-Game-Teams and Enron-Meetings as well as MUC-6 and NewsGroups.

Finally, in the left-hand side of Figure 4, we show the difference between the maximum token-recall gain possible with a MDR-like method and the maximum token-recall gain possible with a SDR-like method. (Here changes are expressed as percentages.) It is clearly preferable to explore MDR-like methods rather than SDR-like for the Mgmt-Game, Mgmt-Game-Teams and Enron-Meetings data, and SDR-like methods for the others. Note that in the particular case of the Mgmt-Game corpus, gains for both the SDR-like and MDR-like methods are limited by the fact that the token recall of the original CRF extractor is high—more than 95%. Enron-Random also appears to offer little scope for recall improvement, perhaps because there is little regularity in the names that appear in it.

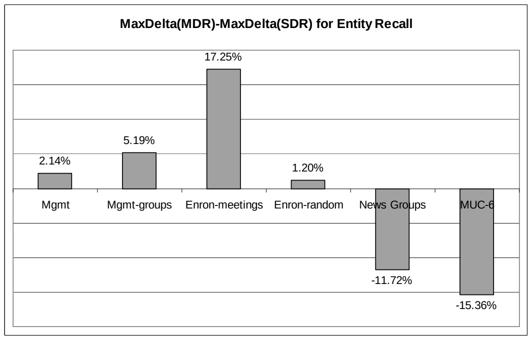
With a little bit of extra bookkeeping, it is possible to perform the same analysis to bound the possible improvements in entity-level recall. The graph on right-hand side of Figure 4 summarizes the result of the entity-level recall analysis. Notice that while the pattern is similar, the possible improvements are generally larger.

The results above emphasize the importance of exploiting repetition of names across multiple documents, as well as within a single document, for entity extraction from email. In the section below, we will describe a method that exploits both single-document and multiple-document repetitions to improve recall.

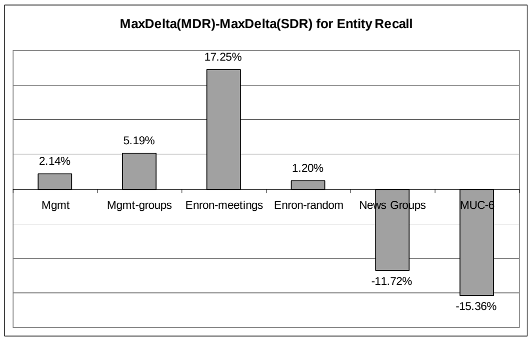
5 Improving Recall With Inferred Dictionaries

Sequential learners of the sort used here classify tokens from each document independently; moreover, the classification of a word w is independent of the classification of other occurrences of w elsewhere in the document. This means that sequential learners cannot easily exploit the "one sense per discourse" constraint. As one example of this, the fact that a word w has appeared somewhere in a context that clearly indicates that it is name (e.g., "Hello Einat, can you help me…") does not increase the probability that it will be classified as a name in other, more ambiguous contexts. One simple and natural approach to correct this problem is to augment the sequential learner with two additional components: one which infers a dictionary of names by analyzing the test documents, and one which matches text against the inferred dictionary to identify additional name occurences that match entries in the inferred dictionary.

In this section we will describe such a technique. First we will describe the matching component, which is dictionary-independent (that is, it could be used with any given dictionary) but task-specific (as it uses normalization rules that work only for person names). Then we will describe several heuristics for inferring dictionaries, all of which are based on taking names extracted from the test set by the learned sequential classifier,



(a) Token recall



(b) Entity recall

Figure 4
Comparing the upper bounds on recall improvements associated with methods that look for terms that re-occur within a single document (SDR) or across multiple documents (MDR). The graph on top is for token-level recall, and the bottom graph is for entity-level recall.

and filtering these names based on certain corpus-wide heuristics.

5.1 Matching names from dictionary

To find names from an existing dictionary, we use a transform-and-match scheme. Each name n is first processed by constructing a family of possible variations of n: as an example, Figure 5 shows the variations created for the name "Benjamin Brown Smith". After all variations of each dictionary name have been created, a single pass is made through the corpus, and every longest match to some name-variation is marked as a name.

Variations from the rightmost column (the initials-only variants of a name) are treated specially. These are marked as a name only if the "inSignoff" feature holds—i.e., if they appear near the end of a message in an apparent signature.

It may be that some name n_1 already identified by the extractor overlaps with a name n_2 identified by the dictionary-matching scheme described above; in other words, the name suggested by the dictionary-matching scheme may conflict with the name suggested by the extractor. In this case, the name n_2 marked by the dictionary-matching scheme is retained, and the name n_1 marked by the extractor is removed.

benjamin brown smith	benjamin-brown-s.	b. brown s.	bbs
benjamin-brown smith	benjamin-b. s.	b. b. smith	bs
benjamin brown-smith	benjamin-smith	b. brown-s.	
benjamin-brown-smith	benjamin smith	benjamin	
benjamin brown s.	b. brown smith	b. smith	
benjamin-b. smith	benjamin b. s.	b. b. s.	
benjamin b. smith	b. brown-smith	b. s.	
benjamin brown-s.	benjamin-s.	brown	
benjamin-brown s.	benjamin s.		

Figure 5
Names variants created from the name "Benjamin Brown Smith"

5.2 Dictionary-filtering schemes

The dictionary-matching scheme works fairly well, but is susceptible to false positives. In preliminary experiments, most of these false positives were created by matching against single-token name-variants, like "bs", "benjamin" and "brown" in Figure 5. One simple way to reduce the number of false positives is to remove some of these single-token name-variants from the dictionary, and one simple way of doing this is by using *extractor confidence*. In other words, it seems natural to filter the dictionary by including in it only multi-token name-variants (which are unlikely to lead to false positives), and single-token name-variants that were generated from a name that was given a high confidence score by the extractor.

Unfortunately, it is non-trivial to assign confidence to an extracted name in general. For CRFs, confidence can be assigned by running the forward-backward algorithm to marginalize out the parts of a document before and after an extracted name (Culotta and McCallu, 2004). For margin-based approaches like VP-HMM, however, there is no obvious entity-level confidence measure. Therefor, in order to assign confidence scores, we trained a *secondary classifier*. The secondary classifier examines the entities extracted by a sequential learner, and then classifies them as correct or false. We investigated a number of training schemes; below we consider a classifier trained using the original voted perceptron method (Freund and Schapire, 1998) using the same features and training data given to the extractor. The rankings produced by this classifier are quite good.

The left-hand graph in Figure 8 shows the result of training and extractor, applying it to the test set, filtering the dictionary created by thresholding the confidence of the secondary classifier, and then applying the matching scheme described above using the filtered dictionary.[4] As expected, including all extractor predictions (confidence threshold zero) severely lowers precision, so F1 performance is also lowered. However, using a dictionary containing only high-confidence extractions improves performance: entity-level

4 More details on this experiment will be given below.

recall increases, and so does entity-level precision (since in matching we also correct boundaries of some partial extractions).

As the threshold is reduced, precision drops. Unfortunately, the drop in precision is very steep, which suggests that a system using this heuristic would be very sensitive to parameter settings. Error analysis shows that the main reason for the sharp drop is certain high-frequency words for which the “one sense per discourse” rule does not hold. For example, “Andrew” is a common first name, and is sometimes confidently (and correctly) recognized as one by the extractor. However, in the Mgmt-Game corpus, “Andrew” is also the name of an email server, and most of the occurrences of this name in this corpus are *not* personal names. The high frequency of the word “Andrew” in the corpus, coupled with the fact that it is only sometimes a name, means that adding this word to the dictionary leads to a large number of false positives.

Since confidence is not sensitive to this property, we investigated three other metrics for filtering the name dictionary. The first metric estimates the degree to which the “one sense per discourse” rule holds for a word—specifically, the degree to which it appears to be used consistently as a name throughout the corpus. This metric is named *predicted frequency* (PF), and is defined as

$$PF(w) \equiv \frac{cpf(w)}{ctf(w)}$$

where $cpf(w)$ is the number of times that a word w is predicted as part of a name in the entire corpus by the extractor, and $ctf(w)$ is the number of occurrences of the word w in the entire corpus. We emphasize that this statistic is based on running the extractor on the test data, not the training data.

Predicted frequency does not assess the likely cost of adding a word to a dictionary: as noted above, terms that occur frequently will lead to more false positives. A number of statistics could be used here; for instance, practitioners sometimes filter a large dictionary by simply discarding all words that occur more than k times in a test corpus. We elected to use the *inverse document frequency* (IDF) of w to measure word frequency:

$$IDF(w) \equiv \frac{\log(\frac{N+0.5}{df(w)})}{\log(N+1)}$$

Here $df(w)$ is the number of documents that contains a word w, and N is the total number of documents in the corpus. Inverse document frequency is often used in the field of information retrieval (Allan et al., 1998), and the formula above has the virtue of being scaled between 0 and 1 (like our PF metric) and of including some smoothing. In addition to bounding the cost of a dictionary entry, the IDF formula is in itself a sensible filter, since personal names will not appear as frequently as common English words.

The third metric combines these two multiplicatively, with equal weights:

$$PF.IDF(w) : \frac{cpf(w)}{ctf(w)} \times \frac{\log(\frac{N+0.5}{df(w)})}{\log(N+1)}$$

PF.IDF takes into consideration both the probability of a word being a name, and how common it is in the entire corpus. Words that get low PF.IDF scores are therefore either words that are highly ambiguous in the corpus (as derived from the extractors' predictions) or are common words in the corpus.

5.3 Evaluation of different dictionary-filtering schemes

From the discussion above, several dictionary-filtering schemes are possible: in addition to using any of the four metrics of extractor confidence, PF, IDF, and PF.IDF, one could also plausibly combine extractor confidence with any of the latter three. In a preliminary exploration of the different approaches, we trained a CRF learner for five iterations on the Mgmt-Game corpus using binary tags and the basic feature set with the focus-word features excluded. This leads to an extractor with a modest F1 value of 70.8, which makes it easier to see the results of dictionary filtering.

We first consider the performance of the PF, IDF, and PF.IDF filters when all extracted names are considered as candidates. In other words, we first extract a set S of names from the test set, using the extractor learned by the procedure above. We then discard all names $n \in S$ such that $PF(n) < \theta$, for $\theta = 0.10, 0.20, \ldots, 1.0$, thus forming a new dictionary S_θ for each threshold. Then we compute the entity-level F1 that results from using dictionary-matching with each S_θ.

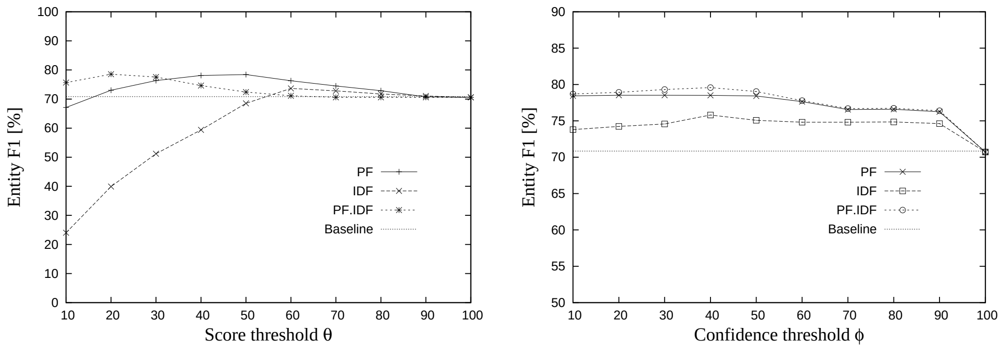


Figure 6
Comparison of the filtering formula components. Left: at varying thresholds for IDF, PF, and PF.IDF, considering all extracted names. Right: setting the threshold levels for IDF, PF, and IDF at optimal levels per the left-hand graph, and varying the confidence threshold.

The left-hand side of Figure 6 shows the result of this experiment, repeated for PF, IDF, and PF.IDF. Both PF and PF.IDF improve over the baseline (of doing no dictionary matching) for most values of θ, and IDF improves over the baseline for a sufficiently high θ.

In the curve, the IDF, PF, and PF.IDF curves peak at thresholds of 0.6, 0.5, and 0.2 respectively. In the right-hand side of Figure 6, we set the thresholds for these functions at these "peak" values, and vary the threshold for extractor confidence. In other words, we compute several initial sets $S^{0.1}, \ldots, S^1$, such that S^ϕ contains all names extracted from the test set that are assigned a confidence greater than ϕ by the secondary classifier. For each value of $\phi = 0.1, \ldots, 1.0$, we then filter S^ϕ by removing all names n such that $PF(n) < \theta_{opt,PF}$, where $\theta_{opt,PF}$ is the "peak" value from the left-hand graph, and then compute the entity-level F1 that results from using dictionary-matching on the resulting set.

The result of this experiment is shown in the right-hand graph of Figure 6. The PF.IDF curve dominates the other two, although the PF.IDF and and PF curves are quite similar.

Figure 7 presents some more detailed results from the experiment. This graph shows token recall, token precision, and token F1 for PF and PF.IDF when used on the dictionary of all extracted names S. As the threshold is decreased, the PF.IDF measure has a much lower drop in precision, but a comparable rise in recall.

Finally, Figure 8 shows the result on token-level performance of varying the extractor confidence ϕ when no subsequent filter is used (on the left) and when the PF.IDF filter is used with $\theta = 0.2$ (on the right). The PF.IDF filter prevents the sharp drop in precision

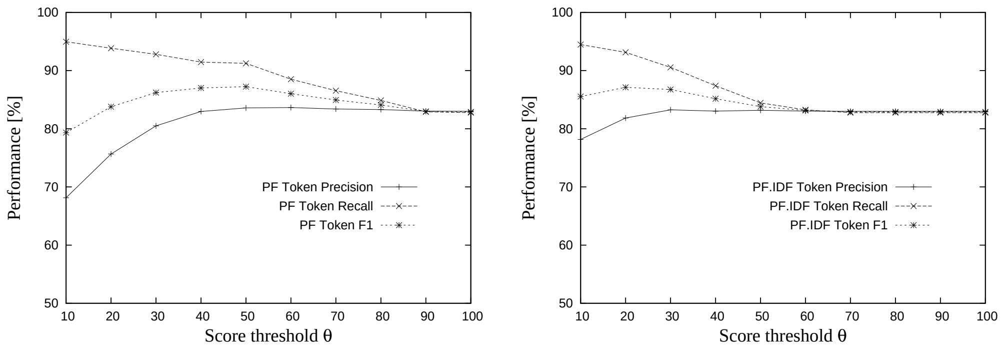


Figure 7
Precision vs. Recall trade-off applying name-matching with PF filter (left) and PFIDF filter (right)

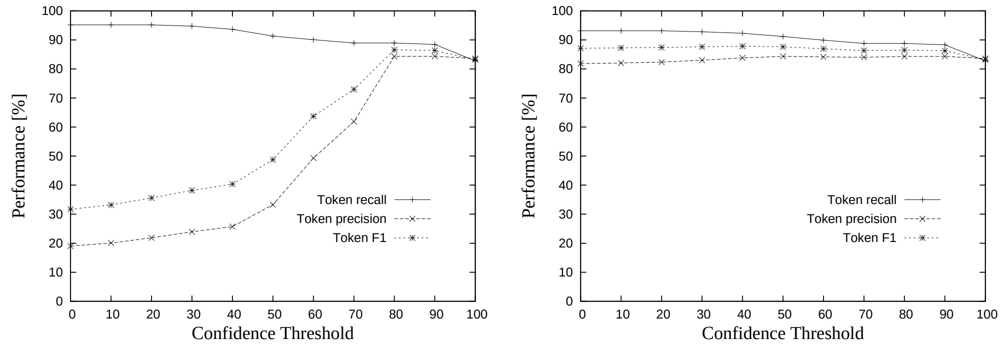


Figure 8
Token precision, recall and F1 behavior as a threshold on extractor confidence is varied. On the left, no filter is applied. On the right, the PF.IDF filter at threshold 20 is applied.

as ϕ is lowered. In fact, F1 performance at $\phi = 0$ is close to best F1 performance reached for any value of ϕ.

Our conclusions from the above experiments are as follows. First, using a dictionary of all extracted names, or even all high-confidence extracted names, is somewhat dangerous: intuitively, the danger is that precision can drop off quite suddenly if a common term that does not obey the "one sense per discourse" rule is added to the dictionary. Second, of the three measures discussed above, the PF.IDF measure seems to be the most robust to parameter settings, and also seems to dominate the other two when parameters are carefully set. PF.IDF at its optimal setting yields the best improvement in recall. Third, the PF.IDF filter with $\theta = 0.2$ provides good performance, even when a threshold of zero is used for extractor confidence.

This suggests the following simple recall-enhancing strategy, which we will evaluate further below.

1. Learn an extractor E from the training corpus C_{train}.

2. Apply the extractor E to a test corpus C_{test} to assign a preliminary labeling.

3. Build a dictionary S_{θ_*} including the names n such that (a) n is extracted somewhere in the preliminary labeling of the test corpus or is derived from an

extracted name applying the names transformation scheme and (b) $PF.IDF(n) > \theta_*$. A reasonable value is $\theta_* = 0.2$.

4.Apply the dictionary-matching scheme of Section 5.1, using the dictionary S_{θ_*} to augment the preliminary labeling, and output the result.

Notice that this procedure does not require using a secondary classifier.

5.4 Experiments with inferred dictionaries

Tables 7 and 8 show results using the method described above. We will initially consider all of the email corpora and both learners in their best configurations, using both the extended feature set. The results are given in terms of relative change, compared to the baseline results generated by the extractors (see Table 5, above).

As expected, recall is always improved. The entity-level F1 (denoted as E-F1) is mostly increased as well, showing that recall is increased more than precision is decreased. The Table also gives F2 measure, which gives higher weight to recall. (This measure reflects the preferences of a more recall-oriented user, for example one who is interested in anonymizing a corpus.) As shown, the change in entity-level F2 is almost always positive, and always higher than the increase in entity-level F1. The method shows modest improvements in F1 for two of the four datasets for both learners, and improvements in F2 for three of the four datasets. The largest improvements are for Enron-Meetings and Mgmt-Game-Teams—the two e-mail datasets shown to have the largest potential improvement from MDR-like methods in Figure 3.

Recall that the extended features already include membership tests in several large name dictionaries, which are not available for all NER tasks. We repeated the same experiments using the basic features. Here performance improvements are larger. An especially large improvement occurs in Mgmt-Game-Teams. As detailed in Section 3.4.2 this dataset was constructed so that the names in the training and test set have only minimal overlap. The substantial performance improvement here shows that repetition of mostly-novel names—some in familiar name-like contexts, and some in ambiguous context—can be used to improve overall performance.

Dataset	VP-HMM				CRF			
	T-F1	T-F2	E-F1	E-F2	T-F1	T-F2	E-F1	E-F2
Mgmt Game	0.3%	0.9%	0.1%	0.9%	0.3%	1.0%	-0.4%	0.6%
Mgmt Game-Teams	0.4%	1.5%	-0.7%	0.7%	1.6%	3.3%	0.8%	2.6%
Enron-Random	-0.3%	2.0%	-4.0%	-1.0%	-0.2%	2.7%	-4.3%	-0.2%
Enron-Meetings	1.0%	3.0%	2.3%	5.3%	3.0%	5.9%	3.9%	7.3%

Table 7
Relative improvement for the results presented in table 5, applying name-matching on models trained with the extended feature set

Dataset	VP-HMM				CRF			
	T-F1	T-F2	E-F1	E-F2	T-F1	T-F2	E-F1	E-F2
Mgmt Game	1.5%	2.8%	2.5%	4.5%	1.6%	3.1%	2.6%	4.8%
Mgmt Game-Teams	7.6%	13.5%	11.4%	20.6%	9.4%	15.8%	15.0%	25.6%
Enron-Random	1.9%	4.5%	-0.5%	3.0%	1.3%	4.2%	0.2%	3.0%
Enron-Meetings	4.9%	8.9%	3.8%	8.1%	3.3%	6.8%	4.9%	7.8%

Table 8
Relative improvement for the results presented in table 5, applying name-matching on models trained with the basic feature set

The datasets considered above are fairly large. In a final experiment, we examined the usefulness of the inferred-dictionary name-matching technique when training data is limited. We split the Mgmt-Game training documents set into four parts, based on email date (thus the parts are not exactly equal in size). We trained the extractor on one, two, three or four days worth of email, which correspond to 25%, 50%, 75%, and 100% of the available training data. The resulting curves for entity-level F1 (on the left-hand side) and entity-level F2 (on the right-hand side) are presented in Figure 9.

The curves show that the name-matching method substantially outperforms the baseline method when less training data is available. In fact, training on one day's worth of email data and applying name-matching gives better performance than training on four day's worth of email, without using name-matching.

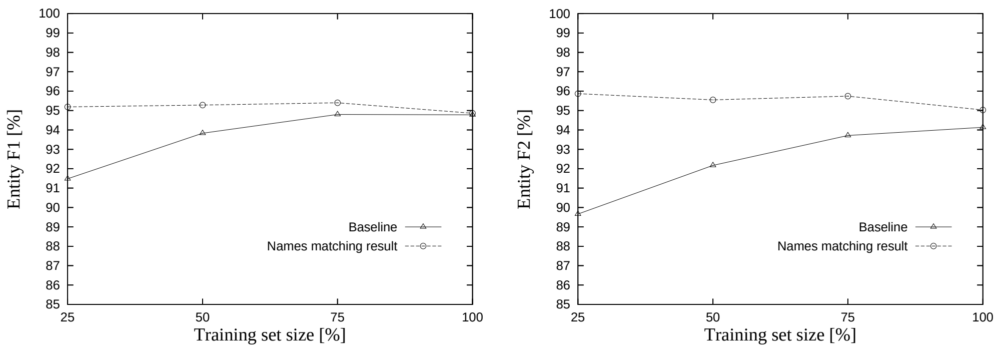


Figure 9
Entity-level F1 and F2 that are output by the extractor (baseline) against the result of inferred name-matching for an increasing amount of training data (% of the Mgmt Game training set)

To summarize the preceding experiments: using our technique, name-matching using a dictionary inferred from test data generally improves recall, even for highly-tuned, state-of-the-art extractors that use as features membership in several large name dictionaries. As implemented, this increase in recall comes at the expense of precision. The technique helps most when names are novel, as in Mgmt-Game-Teams, or dense as in Enron-Meetings; when the features used for learning are not well-tuned, as with the basic feature set; or when there is less training data available.

6 Related Work

Although NER has been well-studied, there have been relatively few studies of NER on informal text. One exception is by Huang (Huang, Zweig, and Padmanabhan, 2001) and Janche and Abney (Jansche and Abney, 2002), who studied the extraction of caller names and phone numbers from voice mail transcripts. Interestingly, position within a message turns out to be an important feature for identifying names in both voice messages and email messages. There has also been some prior work on using sequential learning methods for document analysis of email, specifically identifying signature sections and reply sections of emails (Carvalho and Cohen, 2004).

Our recall-enhancing name-matching method is based on finding all strings predicted to be names somewhere in a document, filtering this set using two simple statistical measurements, and then marking as names subsequent matches to the names from the filtered set. Previously, Stevenson and Gaizauskas (Stevenson and Gaizauskas, 2000) presented a pure dictionary-based method for extracting names from newswire. They evaluated certain ad-hoc rules for filtering dictionaries, for instance using a "probability filter", in which names that occur more frequently as non-names than names in the annotated corpus are removed. The probability filter was also combined with excluding names that appear in a common words lexicon. Although there are many points of similarity to the approach that we describe, the method of Stevenson and Gaizauskas has been tested only for filtering a user-provided dictionary, rather than the far noisier dictionaries formed by running an extractor on the test set. Another difference is that their filter measures ambiguity on the training set, whereas we estimate ambiguity over the test set using a learned extractor. This is an important distinction for datasets such as Mgmt-Game-Teams, in which the sets of names appearing in the testing and training sets are largely different.

Another technique analogous to our PF filter was proposed by Collins (Collins, 2002b) who suggested using, as a feature for NER, the ratio of capitalized to non-capitalized occurrences of a word in an unannotated corpus. Meulder *et al* (Meulder and Daelemans, 2003) also use this feature for learning names from an unannotated corpus. In this work, they identify potential names from the text using dictionaries and capitalization information, and then classify these candidate strings using a learner. In contrast to these works, which rely on capitalization features to identify names, we use capitalization information only as a feature for the extractor. Our dictionary-filtering technique does not use capitalization features. This makes it more applicable to informal texts, where capitalization conventions are often not followed.

There is a long history of improving NER by exploiting repetition within a single document (e.g., (Humphreys et al., 1998; Chieu and Ng, 2002)). Recently, sequential learning methods have been extended to directly utilize information about name co-occurrence in learning the sequential classifier (Bunescu and Mooney, 2004; Raghavan, Allan, and McCallum, 2004). This approach is an elegant solution to modeling repetition within a single document; however, it is not clear that it can scale to modeling multiple-document repetition, which our analysis shows to be important for email.

7 Concluding Remarks

This work applies two recently-developed sequential learning methods to the task of extraction of named entities from email. This problem is of interest as an example of NER from informal text—text that has been prepared quickly for a narrow audience.

We showed that although informal text has different characteristics from its formal text like newswire text, sequential extraction learners such as VP-HMM and CRFs yield reasonable to excellent performance on informal text. Analysis of the highly-weighted features selected by the learners showed that names in informal text have different (and less informative) types of contextual evidence. However, email also has some structural regularities which make it easier to extract personal names. We presented a detailed description of a set of features that address these regularities and statistically significantly improve NER performance on email. These features also appear to make learned extractors more robust to changes in the particular set of names that appear in the test data.

In the second part of this paper, we analyzed the way in which names repeat in different types of corpora. We showed that repetitions within a single document are more common in newswire text, and that repetitions that span multiple documents are more common in email corpora. Additional analysis confirms that the potential gains in recall from exploiting multiple-document repetition is much higher than the potential gains from exploiting single-document repetition.

Based on this insight, we introduced a simple and effective method for exploiting multiple-document repetition to improve an extractor. The method consists of inferring a name dictionary from a test corpus using a trained extractor, filtering the inferred name dictionary, and then matching against names from the filtered dictionary. This approach substantially improves recall, and often improves F1 performance. Performance is improved most when only simpler features are available, or when there is little available training data. The principle limitation of the method is that it requires a reasonable amount of unlabeled test data to be applied.

The principle contributions of the paper are development of new labeled test corpora of email text; evaluation of state-of-the-art NER methods on these corpora; development of novel email-specific feature sets that significantly improve performance; and development of a novel method for exploiting multiple-document repetition of names to improve recall.

A number of additional topics to investigate are suggested by this paper, and may form the basis of future work. We have studied in depth one NER problem involving informal text—person-name extraction from email. It would be valuable to investigate whether other informal-text NER problems show similar patterns—in particular, if the same pattern of multiple-document *vs* single-document named-entity repetition holds for other entity types, and other types of informal text. The method used for filtering dictionaries is potentially quite broadly applicable, and it would be desirable to explore the applicability of this method in other domains, involving other large lexicons and imperfectly-trained extractors.

Acknowledgments
The authors wish to thank Shinjae Yoo for contributions to the early phase of this research. This work was funded by grants from the Information Processing Technology Office (IPTO) of the Defense Advanced Research Projects Agency (DARPA).

References

Allan, J., J. Callan, W.B. Croft, L. Ballesteros, D. Byrd, R. Swan, and J. Xu. 1998. Inquery does battle with trec-6. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*.

Altun, Yasemin, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Hidden markov support vector machines. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*.

Bikel, D. M., R. L. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231.

Borthwick, A., J. Sterling, E. Agichtein, and R. Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Sixth Workshop on Very Large Corpora New Brunswick, New Jersey. Association for Computational Linguistics.*

Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 24(1):543–565.

Bunescu, Razvan, Ruifang Ge, Raymond J. Mooney, Edward Marcotte, and Arun Kumar Ramani. 2002. Extracting gene and protein names from biomedical abstracts. Unpublished Technical Note, Available from http://www.cs.utexas.edu/users/ml/publication/ie.html.

Bunescu, Razvan and Raymond J. Mooney. 2004. Relational markov networks for collective information extraction. In *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning (SRL-2004)*, Banff, Canada, July.

2004. CALO: Cognitive assistant that learns and organizes. http://www.ai.sri.com/project/CALO.

Carvalho, Vitor and William W. Cohen. 2004. Learning to extract signature and reply lines from email. In *Proceedings of the Conference on Email and Anti-Spam 2004*, Mountain View, California.

Chieu, Hai Leong and Hwee Tou Ng. 2002. Named entity recognition: A maximum entropy aapproach using global information. In *Proceedings of the Nineteenth International Confreence on Computational Linguistics*, pages 190–196.

Cohen, William W., Tom Mitchell, and Vitor Carvalho. 2004. Learning to classify email into "speech acts". In *Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

Collins, Michael. 2002a. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Empirical*

Methods in Natural Language Processing (EMNLP).

Collins, Michael. 2002b. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Meeting of the Association for Computational Linguistics*.

Craven, M., D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 2000. Learning to construct knowledge bases from the world wide web. *Artifical Intelligence*, 118((1-2)):69–113.

Craven, Mark and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB-99)*, pages 77–86. AAAI Press.

Culotta, Aron and Andrew McCallu. 2004. Confidence estimation for information extractio. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL).*

Durban, R., S. R. Eddy, A. Krogh, and G. Mitchison. 1998. *Biological sequence analysis - Probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge.

Etzioni, Oren, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-2004).*

Freitag, Dayne. 1998. Information extraction from html: application of a general machine learning approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI.

Freund, Yoav and Robert E. Schapire. 1998. Large margin classification using the perceptron algorithm. In *Computational Learning Theory*, pages 209–217.

Huang, Jing, Geoffrey Zweig, and Mukund Padmanabhan. 2001. Information extraction from voicemail. In *Meeting of the Association for Computational Linguistics*, pages 290–297.

Humphreys, K., R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. 1998. Description of the LASIE-II system as used for MUC-7.

Jansche, Martin and Steven P. Abney. 2002. Information extraction from voicemail transcripts. In *Empirical Methods in Natural Language Processing (EMNLP).*

Klimt, Bryan and Yiming Yang. 2004. Introducing the Enron corpus. In *Proceedings of the Conference on Email and Anti-Spam 2004*, Mountain View, California.

Kraut, R. E., S. R. Fussell, F. J. Lerch, and J. A. Espinosa. 2004. Coordination in teams: evi-dence from a simulated management game. To appear in the Journal of Organizational Behavior.

Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Liu, D. C. and J. Nocedal. 1989. On the limited memory bfgs method for large-scale optimization. *Mathematic Programming*, 45:503–528.

Malouf, Robert. 2002. Markov models for language-independent named entity recognition. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002).*

McCallum, A. K., K. Nigam, J. Rennie, , and K. Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163.

McCallum, Andrew and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of The Seventh Conference on Natural Language Learning (CoNLL-2003)*, Edmonton, Canada.

Meulder, Fien De and Walter Daelemans. 2003. Memory-based named entity recognition using unannotated data. In *Proceedings of the Seventh CoNLL confreence held at HLT-NAACL*, pages 208–211.

2004. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net.

Raghavan, Hama, James Allan, and Andrew McCallum. 2004. An exploration of entity models, collective classification and relation description. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Sha, F. and F. Pereira. 2003. Shallow parsing with conditional random fields. In *In Proceedings of HLT-NAACL*.

Stevenson, Mark and Robert Gaizauskas. 2000. Using corpus-derived names lists for named entities recognition. In *Proceedings of the sixth conference on Applied natural language processing*, pages 290–295.

Sweeney, L. 2003. Finding lists of people on the web. Technical Report CMU-CS-03-168, CMU-ISRI-03-104, Carnegie Mellon University School of Computer Science. Available from http://privacy.cs.cmu.edu/dataprivacy/projects/rosterfinder/.

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196.