High-recall protein entity recognition using a dictionary

Zhenzhen Kou, William W. Cohen and Robert F. Murphy

Center for Automated Learning & Discovery, Carnegie Mellon University, Pittsburgh, PA 15213 U.S.A.

ABSTRACT
Protein name extraction is an important step in mining biological literature. We describe two new methods for this task: semiCRFs and dictionary HMMs. SemiCRFs are a recently-proposed extension to conditional random fields that enables more effective use of dictionary information as features. Dictionary HMMs are a technique in which a dictionary is converted to a large HMM that recognizes phrases from the dictionary, as well as variations of these phrases. Standard training methods for HMMs can be used to learn which variants should be recognized. We compared the performance of our new approaches to that of Maximum Entropy (MaxEnt) and normal CRFs on three datasets, and improvement was obtained for all four methods over the best published results for two of the datasets. CRFs and semiCRFs achieved the highest overall performance according to the widely-used F-measure, while the dictionary HMMs performed the best at finding entities that actually appear in the dictionary—the measure of most interest in our intended application.
Keywords: Protein Name Extraction, Dictionary HMMs, CRFs, SemiCRFs.

1 INTRODUCTION

Searching documents for entities that appear in them is a challenging subtask of information extraction (IE), especially when applied to medical and biological papers (Fukuda et al., 1998; Humphreys et al., 2000; Seki and Mostafa, 2003). Biomedical applications have special types of named entities that are different from those typically addressed by existing named entity recognition systems. These include names of genes, proteins, cell types, and drugs.
Two basic approaches to entity recognition have been described: dictionary-based and context-based. In the dictionary-based approach, a *pattern dictionary* is constructed (Soderland and Lehnert, 1994). When a new document is presented, each textual n-gram in the document is scanned looking for matches to the patterns in the dictionary.
Context-based extractors are usually based on machine learning. The name extraction problem is reduced to classification of individual words (Bikel et al., 1997; Demetriou and Gaizauskas, 2000). First a classifier determines whether each word is part of a named entity, and then the named entity is extracted by identifying the longest sequence of such words. Statistical machine learning techniques, for example hidden Markov models (Bikel et al., 1997), bootstrapping (Demetriou and Gaizauskas, 2000), and CRFs (Ryan and Pereira, 2004), are used to extract names. Machine learning techniques can also be used to construct context-sensitive pattern matching rules (Califf and Mooney, 1999) to extract named entities from text. Of course, these methods do not have to be used in isolation. Humphreys et al. (2000) used grammar rules as well as a lexicon to tag protein names.
Each approach has advantages and disadvantages when applied to protein name extraction. Dictionary-based extractors typically have low recall, unless they are coupled with a soft-matching scheme that handles variant entity names. Dictionary-based extractors also go out of date when the dictionary changes. In principle they can be updated by loading a new dictionary, but in practice, manual curation of a new dictionary is usually required. (For instance, our dictionary contains the entity AT as a protein name; if case-folding is allowed and this entity is not removed, then it would match the common word “at”). Context-based approaches do not in principle need updating when the set of entities changes. However, learned extractors do depend on the "dictionary" of entities available at training time, and it is unclear how they would perform on test sets containing a different distribution of entities—hence they too may go out of date over time.
A number of recent studies have evaluated different approaches specifically to recognizing protein names in MEDLINE abstracts. Franzén et al. (2002) described the YAPEX system, which achieved an F-measure of 67.1% on a dataset of 200 labeled abstracts. Kazama et al. (2002)

* Correspondence should be addressed to Zhenzhen Kou, zkou@andrew.cmu.edu

compared the performance of Support Vector Machines (SVMs) and MaxEnt using the larger GENIA dataset, and found that SVMs gave better precision and F-measure, while MaxEnt gave better recall. Bunescu et al. (2004) compared the performance of a dictionary based approach, a rule learning system, boosted wrapper induction (BWI), SVMs, and MaxEnt on a dataset of over 700 abstracts. They concluded that machine learning approaches using SVMs and MaxEnt are able to identify protein names with higher accuracy than the other approaches. More recently Ryan and Pereira (2004) described an approach using conditional random fields (CRFs) with lexicon features, and achieved an overall F-measure of 82% on the BioCreative evaluation dataset.

In this paper we describe two new methods for recognizing protein names in abstracts. Our work is part of a larger system for extracting information from both images and text in journal articles (Murphy et al., 2004). This system, SLIF, creates a searchable database by mining on-line papers for fluorescence microscope images that show the subcellular localization of a protein. SLIF then analyzes the images, and associates them with the proteins and cell types in the accompanying caption. Queries to this database are expected to request information about the localization of known proteins (for instance, "find v-SNARE proteins that appear to localize to the early Golgi"). For queries such as the one above, recognizing entities that cannot be matched to the list of known proteins is of little interest; therefore, we have focused on *recognizing entities from a fixed list* which may change over time. Also, recall is more important than precision, since the end-user can filter out false positives with additional search constraints.

We therefore developed a novel learning method, dictionary HMMs (Dict-HMMs), which combines a dictionary with a hidden Markov model (HMM) to perform a soft match of phrases in text to entries in a dictionary. Dict-HMMs *learn* how to match words in a large uncurated dictionary. Only a small amount of training data is needed, and the learner is *robust*—meaning that if the training data is slightly different from the target set, the learner can still do reasonably well.

To evaluate the Dict-HMMs approach, we compared its performance to MaxEnt, CRFs, and the recently-proposed semiCRFs, on datasets from different sources. Using a large dictionary from PIR-NREF[1], we obtained an improvement over the best published results on two of the datasets.

2 ALGORITHMS

2.1 Text preprocessing and tokenization

Our algorithm begins by removing stop words, using the list of Seki and Mostafa (2003), and converting the remaining text to tokens. This tokenization is very important to the performance of Dict-HMMs, since there are many surface clues that indicate protein names. We used the rules of Ryan and Pereira (2004) to transform the original words, except that we also added rules for the suffixes "–in" and "–ase", and also new rules indicating the mix of punctuation and characters. The rules used for token transformation are listed in Table 1. (For the other learning methods, these transformation rules were used as features, as discussed below.)

Table 1. Tokenization templates

Rules/templates	Example Input	Example Result
Initial Caps	There	Aa
All Caps	CRF	AA
Caps mix-case 1	SthSth	AaAa
Caps mix-case 2	sthSthsth	aAa
Caps mix-case 3	sthSth	aA
Caps mix-case 4	Sth_sth_Sth	AaA
Char digit mix 1	Sth-123	A-Da
Char digit mix 2	Sth-123-Sth	Aa-Da
Greek letter	alpha	Roma
Suffix	-in, -ase	Sfx
Single Digit	1	DNum
Double Digit	12	DDNum
More Digits	123	DDDNum
Caps+Punctuation1	Aname,sth	Punct_Name
Caps+Punctuation2	aname,sth	Punctname

2.2 Dictionary HMMs

Markov models are mathematical models of stochastic processes that generate random sequences of outcomes according to certain probabilities. An HMM is one in which a sequence of emissions is observed, but the sequence of states the model goes through to generate the emissions is not known. An HMM contains the following elements:

- a set of states $\{S_1, S_2, ..., S_N\}$;
- an alphabet, which defines the set of possible outputs, or emissions, $\{o_1, o_2, ... , o_M\}$;
- an transition matrix A, where A[i, j] is the probability of a transition from state S_i to state S_j;

[1] http://pir.georgetown.edu/pirwww/search/pirnref.shtml

- an emission matrix B, where B[i, k] is the probability of emitting symbol o_k given that the model is in state S_i; and
- a vector π, where π_i is the initial probability of state S_i.

A simple HMM for protein name extraction might be the state automaton shown in Figure 1. Models of this type have been used for some named entity recognition problems (Bikel et al., 1997).

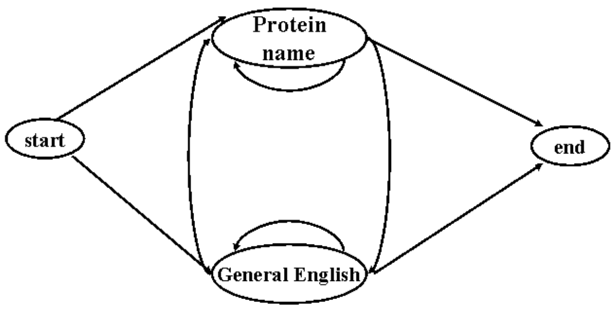


Figure 1. A simple HMM for protein name extraction

Our approach utilizes entries in a protein name dictionary to determine the structure of the HMM model. The new HMM structure is shown in Figure 2. The state *GE* represents *General English*, which corresponds to non-protein text. Each sequence of states $S_{i,1,\ldots,}S_{i,m_i}$, which we will call a *protein path*, corresponds to one protein name from the dictionary—the entry with tokens $a_{i,1,\ldots,}a_{i,m_i}$ and length m_i. This correspondence is enforced by assigning $S_{i,j}$ a high probability of emitting $a_{i,j}$ and a high probability of transitioning to $S_{i,j+1}$. For instance, a token name like "v-SNARE Snc 2" would be associated with a length-three path in the HMM, where the first state S_{i1} is likely to emit the token "v-SNARE". To allow soft matches to entries in the dictionary, the transition matrix also allows "jumping head" from state $S_{i,j}$ to $S_{i,j+k}$ or "looping" from state $S_{i,j}$ back to $S_{i,j}$, as indicated by the grey edges in Figure 2. (For clarity we show only some grey edges) Each path in the HMM is thus similar to a *profile HMM* (Eddy, 1998).

This HMM will be very large. To reduce the number of parameters that must be estimated, we severely constrain the transition and emission matrices A and B. In the following paragraphs in this section, we describe more precisely the dictionary HMM by specifying its structure, alphabet, and emission and transition probabilities.

Defining the paths in dictionary HMMs

For reasons of efficiency, we never construct a complete dictionary HMM: instead, for each text that requires analysis, we will build a smaller HMM that contains only the most relevant protein paths. To select the paths included in a document-specific HMM, we first go through the text to be analyzed, and find all the *relevant entries*—i.e., dictionary entries which contain some word appearing in the text. Using a dictionary with 500,000 protein names, there will typically be a few hundred relevant entries for a 300-word abstract. The set of relevant entries is further reduced using as follows. First, adjacent tokens are grouped into *clusters* if they appear together in some protein name in the dictionary. Quite often tokens in a cluster appear together in several protein names. To further reduce the number of paths, for each cluster, we examine every relevant protein name *P*, and compute the *score* of *P* to the cluster. Here *score=hits(P)/length(P),* where *hits(P)* is the number of cluster tokens that match a token in *P*, and *length(P)* is the

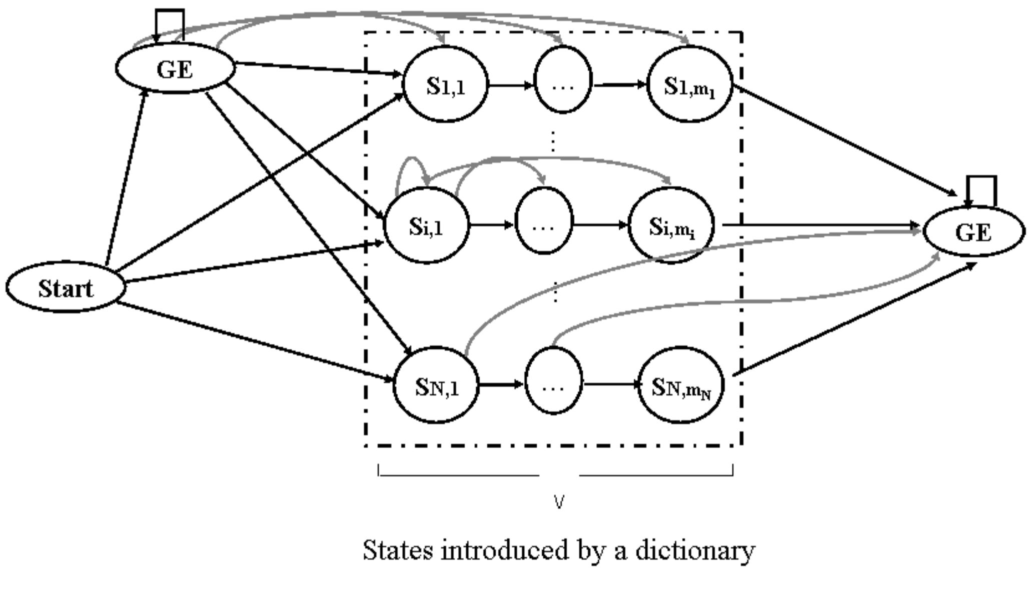


Figure 2. Integrating a dictionary into an HMM

number of tokens in *P*. Finally, we select for each cluster the single protein that has the highest score (breaking ties randomly), and add only that protein's path to the HMM. For example, if the words 'signal recognition particle' formed a cluster, and the relevant protein names were 'signal recognition particle 54K protein' and 'signal recognition particle protein', then only the latter entry would be selected, to generate one path in the Dict-HMM.

Alphabet for the dictionary HMMs

The set of observations include the tokens in the training set, the tokens in the dictionary, and a start symbol (indicating the start of observation sequence). With typical amounts of training data and our large dictionary, there is an imbalance in the number of tokens appearing in each source: the number of distinct tokens in the dictionary is more than 300,000, about 100 times the number of tokens in the training set. This leads to certain problems in smoothing frequency estimates. We thus divide the tokens into 3 categories: tokens only appearing in non-protein text; tokens only appearing in the dictionary, and tokens appearing in both. We then randomly sub-sample the dictionary-only tokens to obtain a smaller set of observations.

Initial probability and transition matrix

The initial probability of state *GE* is estimated from the training data. After estimating $\pi(GE) = \pi_0$, we distribute the remaining possibility among the first states in the *N* paths, i.e., we let $\pi(S_{i,1}) = (1-\pi_0)/N$, where *N* is the number of paths.

The transition probabilities are defined by equation (1) ~ (4) below.

$$\Pr(S_{i,j+k} \mid S_{i,j}) = \frac{\alpha^k}{C} \qquad (1)$$

$$\Pr(GE \mid S_{i,m_i}) = 1 \qquad (2)$$

$$\Pr(GE \mid GE) = \gamma \qquad (3)$$

$$\Pr(S_{i,k} \mid GE) = \frac{\beta^k}{Z}\frac{(1-\gamma)}{N} \qquad (4)$$

Equation (1) defines the transition probability from state $S_{i,j}$ to a following state $S_{i,j+k}$ in a path. The parameter $0 < \alpha < 1$ allows "jumping" but assigns a higher probability to non-jumps and shorter jumps. Equation (2) forces the last state in a path to transition to the *GE* state. In Equation (3), γ is the probability of a transition from *GE* to *GE*, which is estimated from training data. Equation (4) defines the transition probability from state *GE* to state $S_{i,k}$ in a path. The factor $(1-\gamma)/N$ means the probabilities for the transition from state *GE* to states in paths (which should sum up to $1-\gamma$ according to Equation 3) is distributed equally among all the *N* paths. The parameter $0 < \beta < 1$ forces a higher probability to transitions from GE to the first state in a path, but allows a smaller probability of jumping to a state deeper in the path. The parameters C and Z are normalization factors.

Emission matrix

The emission matrix is defined by the following models. First, *Pr(w*$_i$*|GE),* the probability of state GE emitting a token w_i, is estimated from the training data. For the states $S_{i,j}$ in a path, we divide the tokens it can emit into three categories: the tokens a_i in the corresponding protein name entry; tokens only appearing in GE; and tokens appearing in the dictionary sample, excluding the a_i*'s*.. Probabilities of state $S_{i,j}$ emitting symbols in these three categories sum up to 1-ε-δ, δ, and ε, respectively, as defined by Equations (5)-(7):

$$\Pr(a_{i,j} \mid S_{i,j}) = (1-\varepsilon-\delta)/m_i \qquad (5)$$

$$\Pr(w_l \mid S_{i,j}) = \delta \cdot \Pr(w_l \mid Dictionary) \qquad (6)$$

$$\Pr(w_l \mid S_{i,j}) = \varepsilon \cdot \Pr(w_l \mid GE) \qquad (7)$$

In Equation (5), $a_{i,j}$ is a token in the corresponding protein name entry, and the total probability 1-δ-ε is divided by m_i. This means that states in one path have the same probability of emitting any of the tokens $a_{i,1}...,a_{i,mi}$, so the order of the tokens in a compound word is not important. In Equation (6), w_l is a token appearing in the dictionary, excluding the a_i*'s*, and in Equation (7), w_l is a token appearing only in *GE*. The parameters $0 < \varepsilon, \delta < 1$ control the amount of variation allowed in protein names.

Table 2 summarizes the parameters in our model.

Smoothing

During testing we often encounter words that have not been seen during training: hence Equations (5) ~ (7) for emission probabilities need to be smoothed, by allowing novel tokens to appear with some probability. We used Good-Turing smoothing (Gale and Sampson, 1995): i.e., the frequency of tokens that only appear once in the training set is used to predict the total emission probability of unknown words.

Learning the parameters of the model

Above we have defined the structure of the Dict-HMMs, as well as the transition and emission probabilities. We have also described how some of the parameters are set: specifically, the initial probabilities π, the parameter γ, which governs transitions from *GE* to *GE*, and the probabilities *Pr(w|Dictionary)* and *Pr(w|GE)* are all learned from the training data and the dictionary. It remains to learn the transition-matrix parameters α, β, and the emission-matrix parameters δ and ε.

Table 2. Parameters in the model and their functions

Parameter	α	β	γ	ε	δ
Function	Controls transitions from one state $S_{i,j}$ to another state $S_{i,j+k}$ in a path	Controls transition from state GE to a state in a path $S_{i,k}$	Controls transitions from GE to GE	Controls probability of a path $S_{i,j}$ emitting a symbol in GE	Controls probability of a path state $S_{i,j}$ emitting a non-path dictionary symbol

To learn these parameters we use EM, and more specifically a variant of the usual Baum-Welch method. In the usual Baum-Welch method, the emission matrix B and transition matrix A are calculated in the M-step: however there is no guarantee they will follow our assumptions. Therefore after each M-step we compute the best set of values for α, β, δ and ε from A and B by fitting the models associated with each equation to the values associated with A and B. We then re-calculate A and B using these parameters, thus forcing A and B to follow the constraints imposed by equation (1) through (7). We stop updating the parameters when the likelihood of the observed sequence converges.

Because the structure of dictionary HMMs is very dependent on the tokens present in the test data, we run the Baum-Welch algorithm on the *test* sequence, not on the training data, in our experiments. This is feasible since the Baum-Welch algorithm assumes the data are unlabelled.

2.3 Improving the Dict-HMMs

Two additional variations of the Dict-HMM method were evaluated. One introduces a boosting-like strategy into the protein name finding process, and the other adds more states into the Dict-HMM structure.

In the standard Dict-HMMs, all the relevant paths are integrated into one HMM and protein names are extracted using this HMM. If the optimal parameters to extract protein name are different from those for another, the HMM may perform sub-optimally. To reduce the chance of this sort of interaction among paths, we used the following strategy.

(1) Build a Dict-HMM based on a test sentence. If no relevant paths can be found, end the iteration. Otherwise go to step (2).

(2) Learn the parameters in the model with EM, and use the Viterbi algorithm to calculate the optimal state sequence. Then find the *single* protein path with the highest likelihood and report it.

(3) Remove the protein name extracted in step (2) from the sentence. Go to step (1) using the reduced test sentence.

This strategy thus extracts the single most likely protein name at each iteration, and ends when no more protein names are found.

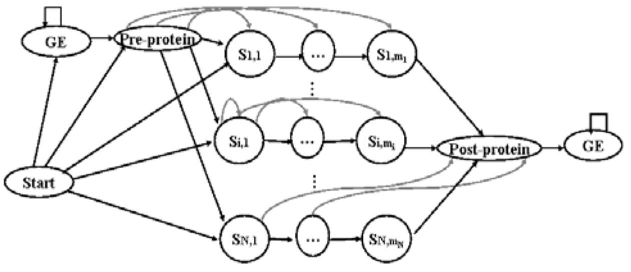


Figure 3. A Dict-HMM with pre- and post-protein states

The second variation was to add to the Dict-HMMs additional states, which include more information about the context surrounding a protein name. We added a *pre-protein* state and a *post-protein* state into the model, thus creating the HMM structure shown in Figure 3.

2.4 Feature calculation

The remaining learning algorithms classify sequences of feature vectors, rather than sequences of tokens. Rather than transforming words in special tokens, as was done for the Dict-HMM, we implemented features (detectors) for the corresponding regularities. For example, there are two features associated with the rule of 'has suffix -in' for MaxEnt, one of which is:

$$f_i(x,y) = \begin{cases} 1 & \text{if current token ends with -in} \\ & \text{\& x is labelled as protein} \\ 0 & \text{otherwise} \end{cases}$$

Table 3. Comparison of methods for protein name recognition. All values are averages over 10-fold cross-validation.

	Precision / Recall / F-measure (%)		
	U. of Texas	GENIA	YAPEX
Best published results	73.4 / 47.8 / 57.9 (Bunescu et al., 2004)	49.2 / 66.4 / 56.5 (Kazama, et al., 2002)	67.8 / 66.4 / **67.1** (Franzén, et al., 2002)
Dictionary-based algorithm from Bunescu et al (2004)	62.3 / 45.9 / 52.8	-	-
MaxEnt	87.2 / 57.3 / 69.1	67.3 / 65.4 / 66.2	69.3 / 58.1 / 63.2
CRFs	83.5 / 66.1 / 73.8	75.0 / 67.6 / 71.1	76.0 / 59.5 / 66.7
SemiCRFs	83.1 / 66.8 / **73.9**	74.8 / 68.3 / **72.3**	76.1 / 58.9 / 66.1
Dict-HMM	46.0 / 69.2 / 55.2	44.8 / 70.1 / 54.7	42.4 / 64.1 / 51.0
Dict-HMM + boosting-like method	49.8 / 74.3 / 59.6	48.3 / 73.9 / 58.5	45.1 / 69.7 / 54.8
Dict-HMM + additional states	51.8 / 72.3 / 60.4	51.3 / 72.4 / 60.1	45.1 / 65.7 / 53.5

(a) Evaluation for all labeled protein names

	Precision / Recall / F-measure (%)		
	U. of Texas	GENIA	YAPEX
CRFs	78.3 / 42.2 / 54.9	71.2 / 45.1 / 55.2	71.8 / 40.5 / 51.8
SemiCRFs	78.0 / 43.1 / 55.5	72.5 / 44.7 / 55.3	72.9 / 39.9 / 51.6
Dict-HMM	47.3 / 67.8 / 55.7	45.0 / 68.7 / 54.4	43.1 / 63.8 / 51.4
Dict-HMM + boosting-like method	50.6 / 73.1 / 59.8	48.9 / 72.0 / 58.2	45.8 / 67.9 / **54.7**
Dict-HMM + additional states	52.3 / 71.0 / **60.2**	52.1 / 70.8 / **60.0**	46.0 / 64.6 / 53.7

(b) Evaluation for protein names with TFIDF similarity score > .9

In addition to such hand-coded features, we also used part-of-speech (POS) tags (from Brill's POS tagger[2]) as features. We also included, for each token *x*, the output of the detectors and POS tags for the three tokens to the left and the right of *x*.

2.5 Maximum Entropy

Maximum Entropy is widely used for inducing probabilistic tagging (Ratnaparkhi, 1996; McCallum, Freitag and Pereira, 2000). Maximum entropy gives a probability distribution of a possible tag *y* given a token *x* $p(y \mid x)$:

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\sum_i \lambda_i f_i(x, y))$$

In this definition, each feature $f_i(x, y)$ is expressed as a binary function based on the current token *x* and its proposed classification *y*, λ_i is the corresponding feature weight, and *Z(x)* is a normalization factor.

2.6 Conditional Random Fields

Conditional Random Fields are another probabilistic tagging model (Lafferty, McCallum and Pereira, 2001). CRFs give the conditional probability of a possible tag sequence $\mathbf{y} = y_1, \ldots y_n$ given the input token sequence $\mathbf{x} = x_1, \ldots x_n$:

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{\exp(\sum_j \sum_i \lambda_i f_i(s_j, \mathbf{x}, j))}{Z(\mathbf{x})}$$

In this definition, each f_i is a function that measures a feature relating the state s_j at position *j* with the input sequence around position *j*, λ_i is the corresponding feature weight, and *Z(*x) is the normalization factor.

2.7 SemiCRFs with dictionary features

Two recent papers (Cohen and Sarawagi, 2004; Sarawagi and Cohen, 2004) compared a number of methods for using dictionaries with CRF-like learning methods. The best results were obtained with a new learning method called semiCRFs. SemiCRFs construct and attach classifications to *subsequences* of a document, rather than to tokens. Since features can measure the properties of these subsequences, a feature measuring the similarity between a candidate seg-

[2] http://www.cs.jhu.edu/~brill/RBT1_14.tar.Z

ment and the closest element in a dictionary can be introduced. We used TFIDF or *cosine similarity* as the sole similarity measurement in our work (Cohen, et al., 2003), and otherwise followed the implementation of Sarawagi and Cohen (2004). To our knowledge, semiCRFs have not been previously evaluated for recognizing protein names. We used Minorthird[3] for the implementation of MaxEnt, CRFs and semiCRFs.

3 EXPERIMENTAL RESULTS

3.1 Dictionary and evaluation datasets

Our dictionary was constructed by extracting the 'protein name' field from the PIR-NREF database. The extracted dictionary contains nearly 500,000 protein names. We used three datasets to evaluate our methods. The University of Texas, Austin dataset[4] contains 748 labeled abstracts; the GENIA dataset[5] contains 2000 labeled abstracts; and the YAPEX dataset[6] contains 200 labeled abstracts.

3.2 Comparision of methods

The performance of Dict-HMMs, MaxEnt, CRFs and semiCRFs were compared for the three datasets. Table 3(a) shows results for all methods, along with published results for the same datasets. With respect to F-measure, the CRF variants improve over the best previous performance on two of the three datasets, and are competitive on the third (YAPEX). The Dict-HMM has lower F-measure performance, but unlike the other methods, appears to emphasize recall over precision.

Bunescu et. al. (2004) explored a wide range of techniques for combining dictionaries and machine learning techniques on the U Texas dataset. One of these, a MaxEnt method that uses a "dictionary tagger", achieved the previous best result. The dictionary tagger used a set of hand-coded generalization rules to convert entries in a dictionary to "canonical" forms, and then tagged a word sequence as a protein name only if it is matched a known "canonical" protein name. Their dictionary was constructed by extracting protein names from Human Proteome Initiative of EXPASY and Gene Ontology Database. Their results are compared to our Dict-HMMs in Table 3(a). The comparison reveals that our Dict-HMM approach is competitive: it has a lower precision but higher recall than Bunescu et al's dictionary lookup algorithm, and achieves a slight improvement in F-measure.

On two of these three problems, semiCRFs—which make use of dictionary information as a feature—improve over conventional CRFs. However, the gains are modest. This is consistent with previous observations that, as measured by F1, the performance gain from dictionary features is largest for small training sets (Sarawagi and Cohen, 2004). This may be because for larger training sets, the most common protein names will be seen in the training data.

Table 3(a) also shows results for enhanced Dict-HMMs. With the boosting-like strategy, the F1-measure is improved by about 4%, the recall is improved by about 5%, the precision is improved by about 3%. With the additional states, the F1-measure is improved by about 4%, the recall is improved by about 2%, and the precision is improved by about 5%. With either of these improvements, the F1-measures for the U. Texas and GENIA datasets are higher than the best previously published results, but still lower than our implementations of MaxEnt and CRFs.

Performance on dictionary entities

Though widely-used, F-measure is often not the best performance measure for specific applications. In our application, we are primarily concerned with finding protein names that can be matched to a known protein from the dictionary. To explore performance with respect to this goal, we calculated the TFIDF similarity score between each extracted protein name and the closest entry in the dictionary. Not surprisingly, the Dict-HMMs method finds only proteins with high similarity scores, while the CRF-based methods do not. For instance, we observed that on the U Texas data, the lowest similarity score for the Dict-HMMs was 0.89, while 26% of the names extracted by CRFs had a similarity score less than 0.89. In our particular application, these "novel", dissimilar proteins are of less interest.

As a quantitative measure of performance in finding dictionary proteins, we calculated precision, recall and F-measure considering only those extracted protein names with a similarity score 0.9 or higher. This is shown in Table 3(b). According to this measure, performance is similar for the CRFs and Dict-HMMs methods, but CRFs had higher precision while Dict-HMMs had higher recall. The enhanced Dict-HMMs achieved the best performance according to the measure with TFIDF scores. For applications such as SLIF, in which non-dictionary entities have less benefit, the Dict-HMM is thus the preferred method.

Learning curve

Experiments were carried out to see how the performance of CRFs and Dict-HMMs depends on the size of training dataset. The 2000 abstracts in the GENIA dataset were split into a testing set of 300 abstracts and training datasets of between 50 and 500 abstracts. The curves for the F-measure, precision and recall are shown in Figure 4, averaged over 10 repetitions. Dict-HMM has comparable recall and precision, even with very small training sets. CRF also learns surprisingly fast. For small training sets, it is still true that the

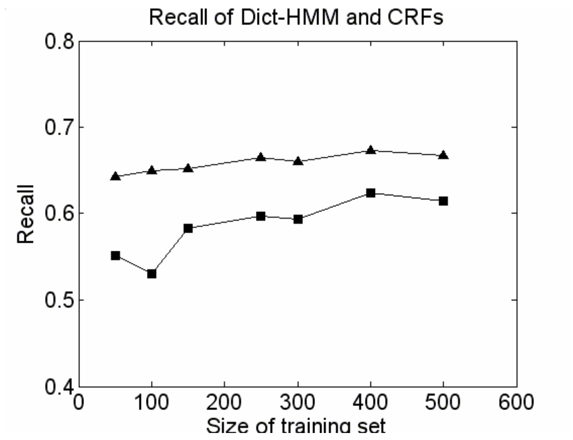
[3] http://minorthird.sourceforge.net/

[4] ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/proteins.tar.gz

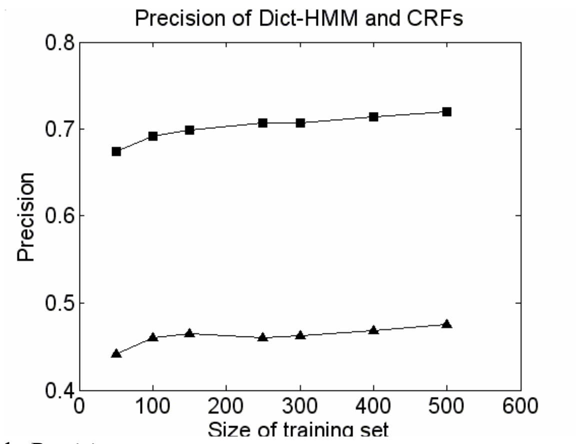
[5] http://www-tsujii.is.s.u-tokyo.ac.jp/~genia/topics/Corpus/posintro.html

[6] http://www.sics.se/humle/projects/prothalt,

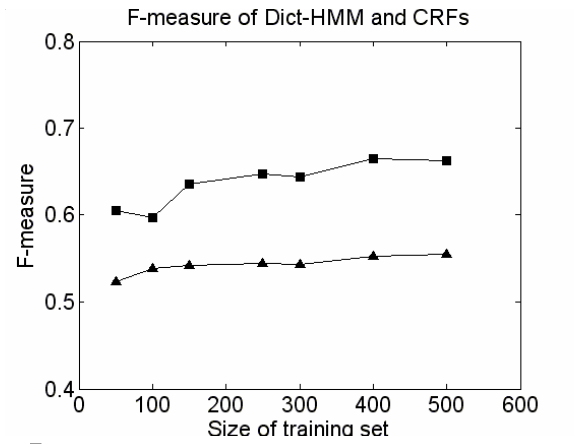
DictHMM has higher recall (and lower precision) than CRFs.



a. Recall



b. Precision



c. F-measure

Figure 4. Dependence of various measures of performance on training set size for Dict-HMMs (▲), and CRFs (■).

Robustness

In practice, it is not easy to get labeled data. Therefore it is important for an algorithm to be able to work well when trained on one dataset and tested on another slightly different dataset. We refer to this property as *robustness*. Experiments were carried out to test the robustness of CRFs and Dict-HMMs. Each of the three datasets were split into a separate training set and a testing set, and a merged training set was obtained by mixing the three training sets. This merged training set was used to learn a model and then the model was applied to the testing sets from each source. The performance (for all protein names) is summarized in Table 4. CRFs and Dict-HMMs performed comparably.

Table 4. Robustness of Dict-HMMs and CRFs.

	F-measure (%)		
	U. Texas	GENIA	YAPEX
CRFs	45.8	63.6	45.0
Dict-HMMs	49.9	50.3	44.3

4 CONCLUSION AND DISCUSSION

Protein name recognition is recognized to be a challenging task. In this paper, we evaluated two new learning methods which make use of large dictionaries. One, Dict-HMM, represents a dictionary as a large hidden Markov model with many shared parameters, and learns to set those parameters to optimize the set of "soft" matches that are recognized. The second, semiCRFs, learns a semi-Markov variant of a conditional random field that uses distance of a phrase to a dictionary entry as a feature.

We compared the performance of Dict-HMMs, semiCRFs, MaxEnt and ordinary CRFs on three test datasets. For two of the datasets, we obtained better F-measure performance than the best previously published. The two CRFs variants also gave comparable results to the best previously obtained for the third dataset. While CRFs or semiCRFs gave the best performance according to F-measure, the boosted Dict-HMMs had a significantly higher recall than any previous system, and also extracted only names that are highly similar to ones in the dictionary. If “novel”, non-dictionary names are discounted, as in our intended application Dict-HMMs have the best performance overall.

Dict-HMMs have two additional advantages: parameters for the model can be learned from a small amount of training data, and the Viterbi path through the dictionary HMM can help identify the best-matching record in the dictionary.

There is still much room for improvement in systems for addressing the protein recognition problem. We are currently exploring using filtering rules, and also an abbreviation finder, in the hopes of improving performance.

5 ACKNOWLEDGMENTS

The work described here was supported in part by research grant 017396 from the Commonwealth of Pennsylvania Department of Health and by NIH K25 grant DA017357-01.

REFERENCES

Bikel, D. M., Schwartz, R., and Weischedel, R. M.. 1997. NYMBLE: A High-Performance Learning Name-finder. Proceedings of the Fifth Conference on Applied Natural Language Processing, Association for Computational Linguistics, pp. 194-201.

Bunescu R., Ge R., Kate R., Marcotte E., Mooney R., Ramani A., Wong Y. W.. 2004. Comparative Experiments on Learning Information Extractors for Proteins and their Interactions. To appear in Journal Artificial Intelligence in Medicine (Special Issue on Summarization and Information Extraction from Medical Documents), 2004.

Califf M. E., Mooney R. J.. 1999. Relational learning of pattern-match rules for information extraction. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), pp. 328-334.

Cohen W. W., Ravikumar P. and Fienberg S.. 2003. A Comparison of String Metrics for Matching Names and Records in KDD Workshop on Data Cleaning and Object Consolidation.

Demetriou,G. and Gaizauskas,R.. 2000. Automatically augmenting terminological lexicons from untagged text. In Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-2000), pp. 861–867.

Eddy, S. R. 1998. Profile hidden Markov models. Bioinformatics, 14:755-763.

Franzén K., Eriksson G., Olsson F., Asker L., Liden P., Coster J.. 2002. Protein names and how to find them. International Journal of Medical Informatics special issue on Natural Language Processing in Biomedical Applications. December, 2002, pp. 49-61.

Fukuda, K., Tsunoda, T., Tamura, A., and Takagi, T.. 1998. Toward information extraction: Identifying protein names from biological papers. In Proceedings of 1998 the Pacific Symposium on Biocomputing (PSB-1998). pp. 707–718.

Gale W. and Sampson G.. 1995. Good-Turing smoothing without tears. Journal Quantitative Linguistics 2,pp. 217–37.

Humphreys, K., Demetriou, G., and Gaizauskas, R.. 2000. Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structures. In Proceedings of 2000 the Pacific Symposium on Biocomputing (PSB-2000). 2000, pp. 502-513.

Kazama J., Makino T., Ohta y., and Tsujii J.. 2002. Tuning Support Vector Machines for Biomedical Named Entity Recognition. In Proceedings of the Natural Language Processing in the Biomedical Domain (ACL 2002), pp. 1-8.

Lafferty J., McCallum A., and Pereira F.. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. 18th International Conf. on Machine Learning, pp. 282-289, 2001.

McCallum A., Freitag D., and Pereira F.. 2000. Maximum entropy Markov models for information extraction and segmentation. In Proc. 17th International Conf. on Machine Learning, pp. 591-598, 2000.

Murphy R.F., Kou Z., Hua J., Joffe M., and Cohen W.. 2004. Extracting and Structuring Subcellular Location Information from On-line Journal Articles: The Subcellular Location Image Finder. In Proceedings of the IASTED International Conference on Knowledge Sharing and Collaborative Engineering (KSCE 2004), pp. 109-114.

Ratnaparkhi A.. 1996. A maximum entropy part-of-speech tagger. In Proc. Empirical Methods in Natural Language Processing Conference, pp. 133--141.

Ryan M. and Pereira P.. 2004. Identifying Gene and Protein Mentions in Text Using Conditional Random Field. http://www.pdg.cnb.uam.es/BioLink/workshop_BioCreative_04/handout/

Sarawagi S. and Cohen W.. 2004. Semi-Markov Conditional Random Fields for Information Extraction. NIPS 2004.

Seki, K., & Mostafa, J. 2003. An Approach to Protein Name Extraction using Heuristics and a Dictionary. Annual Conference of the American Society for Information Science and Technology (ASIST 2003), Long Beach, CA, Oct., 2003.

Sha F. and Pereira F.. 2003. Shallow parsing with conditional random fields. In Proceedings of HLT-NAACL 2003. Association for Computational Linguistics.

Soderland S. and Lehnert W.. 1994. Wrap-up: a trainable discourse module for information extraction. Journal of Artificial Intelligence Research, 2:131--158.