Recognizing Biomedical Named Entities
In the Absence of Human Annotated Corpus

Baohua Gu, Veronica Dahl, Fred Popowich

School of Computing Science, Simon Fraser University
Burnaby, BC, V5A 1S6, Canada
{bgu, veronica, popowich@cs.sfu.ca}

Abstract

Biomedical Named Entity Recognition is an important task in biomedical text mining. Currently the dominant approach is supervised learning, which requires a sufficiently large human annotated corpus for training. In this paper, we propose a novel approach aimed at minimizing the annotation requirement. The idea is to use a dictionary which is essentially a list of entity names compiled by domain experts and sometimes more readily available than domain experts themselves. Given an unlabelled training corpus, we label the sentences by a simple dictionary lookup, which provides us with highly reliable but incomplete positive data. We then run a SVM-based self-training process in the spirit of semi-supervised learning to iteratively learn from the positive and unlabeled data to build a reliable classifier. Our evaluation on the BioNLP-2004 Shared Task data sets suggests that the proposed method can be a feasible alternative to traditional approaches when human annotation is not available.

1 Introduction

Named Entity Recognition (NER) is a key task in biomedical text mining, because biomedical named entities represent concepts of research interest (e.g., proteins/genes/virus/cells). Approaches to the task basically fall into three groups: dictionary-based, rule-based, and supervised-learning-based. The dictionary-based approach assumes the presence of a dictionary of names of target types and identifies names in a text by using exact or approximate string-matching techniques, which makes it suffer from low coverage of the dictionary used. The rule-based approach typically uses handcrafted linguistic rules (or patterns/templates) and seeks named entities by pattern-matching or template-filling. The problem is that good rules require hard work from domain experts, and are not easily adaptable to new domains. In recent years, supervised learning techniques have become dominant, with better performance and adaptability. However, their major drawback is that they need a sufficiently large annotated corpus to build an accurate model.

So far most annotation work is manually done by humans (e.g., domain experts), and the process has proven to be time-consuming and error-prone. For example, the well-known GENIA corpus, debuted in the year 2000 with 500 annotated Medline abstracts (Tateisi et al, 2000) took about 4 years to evolve into its third version, which contains 2000 annotated abstracts. Though it is widely used in biomedical NER as a benchmark as in the BioNLP-2004 Shared Task (Kim et al, 2004), it has been shown that there are considerable inconsistencies in its annotation (Dingare et al, 2004).

While annotated texts are difficult to obtain, unannotated texts (e.g., Medline abstracts) are more readily available. As such, active learning and semi-supervised learning have recently attracted attention. Both make use of unannotated texts to reduce the requirement of annotated texts. However, both methods, among other things, still require humans’ involvement in annotation either at the beginning or during the process of learning.

In this paper, we consider the NER task in a new setting where human annotation is not available.
The idea is to use a dictionary (a list of names), instead of a human annotator, to label sentences, and then employ a technique similar to semi-supervised learning to automatically learn a model from the partially labelled sentences.

The assumption of having such a dictionary is sometimes easier to satisfy than that of having a human annotator. This is the case in the biomedical NER, where many names of proteins and genes can be found in a number of established databases. Although they might have limited coverage, the dictionaries typically are compiled by domain experts and thus are supposedly of good quality.

The difficulty of this new setting lies in that we often cannot completely label all words in a sentence, because some entities in it may not be covered by the dictionary. In fact, after dictionary labelling, we will only have positive examples (i.e., those words matching dictionary entries), but no negative examples (i.e., all other words remain unlabeled because their true labels are not clear). Thus, the problem becomes how to learn from positive and unlabeled examples.

To address this issue, we propose a self-training technique using support vector machines (SVMs) as the base learner. Given the positive and unlabeled examples, we let the SVM iteratively identify reliable (or strong) negatives from unlabeled examples and use its own classification results to teach itself. We evaluated our technique on the BioNLP-2004 Shared Task corpus and obtained encouraging results.

The rest of the paper is organized as follows: We discuss related work in the next section. The details of our method are presented in Section 3. Section 4 describes the evaluation. Our conclusions and future work are discussed in Section 5.

2 Related Work

2.1 Supervised Learning in NER

A number of supervised learning algorithms have been explored in newswire NER, for example, HMM (Miller et al., 1998), Maximum Entropy (Borthwich, 1999), AdaBoost (Carreras et al., 2003), SVM (Mayfield et al., 2003), and CRF (McCallum and Li, 2003). Supervised learning based systems have shown better performance than rule-based systems, e.g., (Zhou and Su, 2002). In biomedical NER, the representatives are SVM (Lee et al., 2003), HMM (Zhou et al., 2004), Maximum Entropy (Lin et al., 2004) and CRF (Settles, 2004). Two best-known shared tasks held in this area are BioNLP (Kim et al., 2004) and BioCreAtive (Yeh et al., 2004), where most participants used supervised learning techniques.

2.2 Using Unlabelled Data in NER

The idea of employing unlabelled data was first introduced to the NLP community by Yarowsky (1995) for the task of word sense disambiguation. Collins and Singer (1999) probably was the first to use unlabeled data for NE classification. Riloff and Jones (1999) proposed a bootstrapping method for NER in web documents by iteratively learning language patterns from unlabelled text. Ando and Zhang (2005) presented a structural learning paradigm for semi-supervised learning, which aims to learn the most predictive low-dimensional feature projection from unlabeled data. Shen et al (2003) studied active learning for NER by using multi-criteria in example selection.

2.3 Unlabeled Data for Text Classification

Much previous work using unlabeled data involved text classification (or document categorization), where semi-supervised learning was studied along two lines. One line of work requires a small set of labelled texts and tries to use a large set of unlabeled texts to augment the labelled set. Blum and Mitchell (1998) proposed co-training which uses two distinct views of each document. Nigam and Ghani (2000) studied the effectiveness and applicability of co-training. Joachims (1999) introduced Transductive SVMs to minimize misclassifications of unlabeled documents.

We did not use the popular co-training algorithm, because it has two conditions: (1) it requires a natural split of the feature set, and (2) each subset of features is sufficient to train a reliable classifier. In the case of biomedical NER, the second condition seems hard to satisfy, because currently the best performance (about 70% F-score) was achieved by using all the available features plus some post-processing heuristics (Kim et al., 2004).

Another line of work addresses practical tasks where only positive (but no negative) examples and unlabeled examples are provided. In order to build a classifier, one has to identify reliable negative examples. Algorithms proposed for this purpose include Spy in S-EM (Liu et al., 2002), 1-DNF in PEBL (Yu et al., 2002), and NB in (Liu et al,
Once reliable negative examples have been selected, an iterative process like that in semi-supervised learning can be applied, for example, Mapping-Convergence of SVM in PEBL (Yu et al., 2002), and NB plus EM in (Liu et al., 2002).

Our work is more similar to the second line of work. However, ours is different in that we use a dictionary to label the text instead of requiring any human involvement. Besides, we use an SVM to identify reliable negatives, and let the SVM teach itself in later iterations. Also, the task we study is NER, which is different from text classification.

To our knowledge, (Jones, 2005) is probably the most similar work to ours, which studied semi-supervised learning for general NER. However, their approach required preliminary chunking of syntactic categories (i.e., noun phrases, verb phrases, and prepositional phrases). In other words, they only considered the potential semantic relations between a noun phrase and its context. If their work can be considered phrase-based, ours is purely token-based in the sense that we do not assume any phrase-chunking as pre-processing. Besides, our technique is evaluated on benchmark NER corpora, while theirs was not.

3 The Proposed Technique

We treat the NER task as a classification problem. Given a sentence, the task is to assign a class label to each word (or token). If multiple classes are involved, we reduce it to binary classification subproblems. As mentioned in the introduction, our technique has two steps: (1) labelling sentences using a dictionary; (2) building a stable classifier by learning from the partially labelled sentences. We assume that a dictionary containing hundreds and even thousands of recognized entity names is not difficult to obtain. In the biomedical domain, for example, a number of established databases, including SwissProt and GenBank\(^1\), are available. In this section, we will describe the details of the two steps as well as some relevant issues.

3.1 Labeling Sentences using a Dictionary

Given a sentence, we label the words in it by performing longest match through the sentence against all entries in the dictionary. Consider the following sentence from the GENIA corpus\(^3\).

**IL-2 gene** expression and **NF-kappa B** activation through **CD28** requires reactive oxygen production by **5-lipoxygenase**.

According to the GENIA annotation, here *IL-2 gene* is a gene, while *NF-kappa B*, *CD28* and *5-lipoxygenase* are all proteins. If the target entity type is protein, and the protein dictionary we have only contains two entries: *NF-kappa B* and *CD28*, then only the 3 words in the sentence will be labelled as positives, while all others as unlabeled. Note that the protein *5-lipoxygenase* is not correctly labelled, because it is not in the dictionary.

After labeling sentences using the given dictionary, we have only positive tokens but not any negative ones. Among the remaining tokens, some could be positive tokens that are not covered by the dictionary, while all the others are those true negative tokens. We take them as unlabelled data.

The labels for the positive tokens are reliable in most cases, except when the token appears in embedded entities or compound nouns. This can be demonstrated by the above example sentence. Actually, the token *IL-2* is also annotated as a protein when it appears independently (e.g., not followed by the word *gene*) in the corpus. However, *IL-2* would be labeled as positive if it were in our protein dictionary. This would result in a contradiction with the GENIA annotation, which only labels *IL-2 gene* as a whole as a gene.

We note that GENIA also labels *IL-2 gene expression* in the same sentence as some entity that relates to *IL-2 gene*. Thus, we speculate that it is not unacceptable to label *IL-2* as an embedded entity, as it appears to us that the *IL-2 gene* is a gene that relates to protein *IL-2*. We attribute such issues to ambiguity resulted from inconsistent annotation, and will leave them to biologists to solve. In our later evaluation, we still take the GENIA annotation as the gold standard. For all those words whose dictionary labels are different from the GENIA annotation, we neglect the dictionary labels and treat the words as unlabeled.

3.2 Choosing the Base Learner

We use SVM as the base learner. As a binary classification algorithm, SVM has shown outstanding performance in many classification problems (e.g.,

\(^1\)http://www.expasy.org/sprot/
\(^3\)http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/
document classification). Besides, it has two salient properties that are highly desirable for our task: (1) it can handle high dimensions and tolerate sparse instance spaces, and (2) for each prediction, it produces the distance from the classification boundary, which can be taken as the confidence of the prediction. The first property allows us to associate hundreds of thousands of features with each word, while the second property helps us to identify reliable negatives from the unlabeled data, which are far from the classification boundary.

Here we do not describe how SVM works, which can be easily found in the SVM literature. The basic idea of how SVM builds a classifier is illustrated in the Figure 1. Note that other classifiers (hyperplanes) may also separate the positive data (marked as “+”) from the negative data (marked as “-”). However, the SVM hyperplane is the one that maximizes the margins between the two classes. The data points on the boundary are called support vectors.

We speculate that the base learner actually can be any classification algorithm as long as it outputs a value that measures the confidence or goodness of its prediction. In case of SVM, this value is the distance of a data point from the hyperplane.

In our experiments, we use the SVM-light package as the implementation of SVM (Joachims, 1999). For efficiency reason, we used the default linear kernel in our experiments, and found that the classification accuracy is acceptable. In the future, we plan to explore other types of kernels.

### 3.3 Selecting the Initial Negatives

Now our task becomes building a classifier from the initial positive set $P_0$ and the unlabelled set $U$ obtained from the dictionary labelling. For this end, we need to select the initial set of negatives $N_0$. We do this as follows:

1. Assign to each token in $P_0$ the class label $+1$;
2. Assign to each token in $U$ the class label $-1$;
3. Build a SVM classifier using $P_0$ and $U$;
4. Classify $U$ with the classifier;
5. Sort the predictions of $U$ by the distance from classification boundary in descending order;
6. Select the bottom $k$ words that are predicted as negatives to form the initial reliable negative set $N_0$, while the remaining are left in the $U$ by resetting the class labels to 0 (meaning unlabelled);

Here $k$ is a parameter which can be set based on the problem. In our experiment, we set it to be equal as the size of $P_0$, which already works well. The initial classifier built in this stage is likely not accurate. We will expand the set $N$ by SVM self-training described in the next section.

### 3.4 SVM Self-Training

Self-training allows the learner teach itself. Given the positive set $P_0$ and the initial negative set $N_0$, following the above notations, we describe the algorithm as follows:

1. let $N = N_0$;
2. Loop while $U$ is not empty:
   1) build a SVM classifier $M$ from $P_0$ and $N$;
   2) classify $U$ using the classifier $M$;
   3) sort the predictions by the distance;
   4) select the bottom $k$ negatives from the predictions and add them to $N$; All others are remained in $U$;
   5) if the model has been stable, break;
3. output the final model $M$;

Note that other stopping criteria can also be used, e.g., the maximum number of iterations. Instead of specifying the parameter $k$, we can also set a predefined cut-off value of distance to select reliable negatives from unlabelled data. However, in our experiments, we observe that this has actually introduced more misclassified negatives into the expanded negative set. Thus, we did not use it.

The self-training process can be illustrated by Figure 2. Though we did not draw the classification hyperplane generated by each iteration, readers can imagine that it locates right in the middle between the positive and the corresponding negative...
tive set. And as the self-training continues, it will approach to the true SVM hyperplane $H$.

Figure 2. Graphical Representation of how the negative set is expanded in the self-training process. The solid points are positives, while the empty points are negatives. The ellipse $P_0$ stands for the set of initial positives, while $N_0$ is the initial set of negatives obtained by taking all unlabeled data as negatives. Ellipses $N_1$ to $N_4$ are the expanded sets of negatives resulted from iteration 1 to 4. $H$ is the true SVM hyperplane.

3.5 Some Relevant Issues

Why is it possible to learn with no negative examples?

Denis et al (2002) gave an explanation from a probability perspective. Using the positive training data, we can estimate the conditional probability of example $x$ given that it is positive: $p(x|+)$. Using the unlabeled data, we can estimate the probability of $x$: $p(x)$. If the probability of the positive class $p(+)$ can be estimated, we can calculate the conditional probability of $x$ given that it is negative: $p(x|-) = [p(x) - p(+) \cdot p(x|+)]/p(-)$, Where $p(-) = 1 - p(+)$.

With the $p(x|-)$, we can do classification using Bayes rule as follows: (1) compute $p(+|x) = p(+) \cdot p(x|+)/p(x)$, and $p(-|x) = p(-) \cdot p(x|-)/p(x)$; (2) choose the one with higher probability as the predicted class.

Why not expand the positive set?

Ideally we hope to expand the positive set as well as the negative set, so that the resulting classifier could approach the one built from fully annotated data. We could do the two expansions at the same time, or at different times (e.g., first expanding the negative set to some predefined size). However, the idea did not work well in our experiments. We tracked the class labels assigned to selected "reliable" positives and negatives during the self-learning. We found that the misclassification rate of the positives is far larger than that of the negatives. Therefore, if we also expanded the positive set, many mislabeled data would be introduced, so that would seriously bias the classifier built in later iterations. This could be related to the nature of the problem: positive words are in themselves harder to identify. How to reduce the misclassification rate of the positives is our ongoing work.

4 Evaluation

4.1 Data Sets

The evaluation was done using the data sets of the BioNLP-2004 Shared Task (Kim et al, 2004). The training data was derived from the GENIA corpus version 3, containing 2000 abstracts. For the shared task, all entities were annotated into 5 classes: protein, DNA, RNA, cell line, and cell type, using the IOB2 notation. The testing data contains 404 abstracts that were annotated for the same 5 classes of entities. In total, there are 492551 and 101039 token occurrences in the training set and the testing set respectively.

4.2 Features

Feature selection is essential to any classification task. Previous studies have found that the orthographic, morphological, affixes, gazetteer, syntactic features are usually useful for biomedical NER. Moreover, the more features are used, the better performance can be expected, while the more computing resources are required. As our goal in this study was not to seek the best features toward the best performance, we only considered the following features in the evaluation, which already gave us comparable performance.

1. Orthographical features are used to capture capitalization, digitalization, and other token formation information. Some of such features and examples are given in Table 1.

2. We use frequent prefixes and suffixes as morphological features. Given all token occurrences in the training set, we count the frequencies of each

---

4 http://research.nii.ac.jp/~collier/workshops/JNLPBA04st.htm
prefix and suffix (3-5 chars long) of a token, and select those occurring more than 3 times as the frequent prefixes and suffixes respectively.

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Examples</th>
<th>Feature name</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1^{st} char_upper</td>
<td>MHC</td>
<td>Has_dot</td>
<td>PU.1</td>
</tr>
<tr>
<td>1^{st} char_lower</td>
<td>Gene</td>
<td>Has_comma</td>
<td>1,25(OH)2D3</td>
</tr>
<tr>
<td>1^{st} char_digit</td>
<td>5-lip</td>
<td>Has_hyphen</td>
<td>peri-kappa</td>
</tr>
<tr>
<td>Last char_upper</td>
<td>ROI</td>
<td>Has_slash</td>
<td>enhancer/promoter</td>
</tr>
<tr>
<td>Last char_lower</td>
<td>c-myc</td>
<td>Has_parenthesis</td>
<td>CCK(B)</td>
</tr>
<tr>
<td>Last char_digit</td>
<td>CD28</td>
<td>Has_plus</td>
<td>K+</td>
</tr>
<tr>
<td>Upper_inside</td>
<td>hEpoR</td>
<td>Roman_num</td>
<td>Type-II</td>
</tr>
<tr>
<td>Lower_inside</td>
<td>PuB2</td>
<td>Greek_num</td>
<td>gamma(c)</td>
</tr>
<tr>
<td>Digit_inside</td>
<td>E1A</td>
<td>Has_others</td>
<td>FY*B, t(8;21)</td>
</tr>
</tbody>
</table>

Table 1. Some orthographic features and examples.

3. Part-of-Speech features have been shown useful in determining boundaries of biomedical NEs. Here we use the PennTreeBank tag set, plus some special tags used in the GENIA corpus.

4. Frequent token features. This feature set is also called gazetteer features. We have observed that this set of features is particularly useful for biomedical NER. We count the frequencies of all tokens in the training set, and take all those words appearing more than 3 times as frequent tokens. Basically, given a list of \( k \) such tokens, for any token to be featurized, if it matches the \( j \)-th token in this list, then we set the \( j \)-th position of the corresponding feature vector as 1, otherwise set it as 0.

As such, we have about 25000 features for each token. In order to capture context clues of a token, we also incorporate the features from its previous 3 tokens and its next 3 tokens (i.e., the context window size is 7). In total, we used 175000 features for each token, in both the skyline system and the prototype system described below.

4.3 Skyline System

We built the skyline system by using all annotations provided in the training set to train a SVM classifier (without any post-processing steps). This is equivalent to assuming that we have a perfect dictionary that can correctly recognize every occurrence of all entities in the training corpus, as well as be able to solve ambiguity arising in case of embedded or overlapping entities. The performance of exact match is shown in Table 2. We expect our proposed technique can approach this ideal performance.

Note that the training and the test corpus are annotated using the IOB2 scheme on five types of entities. In total, we have 11 class labels, namely, B-protein, I-protein, B-DNA, I-DNA, B-RNA, I-RNA, B-cell_type, I-cell_type, B-cell_line, I-cell_line, and O. We converted the multi-class problem into a combination of 11 binary classification problems. Basically, we built a binary classifier for each class label. When predicting the class label for an unseen token, we ran all the 11 classifiers and got 11 predicted values, from which we chose the one having the largest absolute value to be the predicted class label. Though this simple combination seemed naive and can be improved in many ways, it worked well in the task. Thus we also used it in our self-trained SVM classifiers.

4.4 Baseline System

We built the baseline system using purely a dictionary approach. Given a dictionary (a list of entities of all classes), we performed longest match through the test set. We made a 100% coverage dictionary by extracting all the distinct entities from the training set. By randomly selecting a subset from this dictionary, we evaluated the baseline performance under different dictionary coverages.

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Skyline System Precision/Recall/F-score</th>
<th>Baseline System Precision/Recall/F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protein</td>
<td>71.21% / 62.04% / 66.31%</td>
<td>58.24% / 47.03% / 52.04%</td>
</tr>
<tr>
<td>DNA</td>
<td>53.69% / 67.10% / 59.65%</td>
<td>33.33% / 22.81% / 27.09%</td>
</tr>
<tr>
<td>RNA</td>
<td>55.93% / 63.46% / 59.46%</td>
<td>32.20% / 16.96% / 22.22%</td>
</tr>
<tr>
<td>cell_type</td>
<td>57.47% / 80.00% / 66.89%</td>
<td>55.44% / 41.86% / 47.70%</td>
</tr>
<tr>
<td>cell_line</td>
<td>46.20% / 56.48% / 50.83%</td>
<td>32.20% / 29.65% / 30.87%</td>
</tr>
<tr>
<td>[-ALL-]</td>
<td>64.37% / 65.19% / 64.78%</td>
<td>52.72% / 41.04% / 46.15%</td>
</tr>
</tbody>
</table>

Table 2. Performance of the skyline system and the baseline system using the 100% dictionary.

The performance (precision, recall and F-score) of the baseline system with the 100% dictionary is also given in Table 2. Note that the skyline outperforms the baseline by about 0.18 F-score, which reflects the learning ability of SVM given the GENIA annotation and the above featurization. The baseline is almost the same as the baseline used in the BioNLP-2004 Shared Task, while the
skyline system would rank in the middle of all the participant systems in terms of the overall F-score.

4.5 Results and Discussion

We implemented the technique described in Section 3. To simulate real world scenarios where a perfect dictionary is hard to obtain, we used a random subset of the above-mentioned 100% dictionary to label the training set. By adjusting the sampling size, we tried 10%, 20%, up to 90% of the perfect dictionary. With the dictionary of a given size, we labelled all sentences in the training set from which a SVM model was built by the self-training process. The model was then evaluated against the testing set. The base learner was the SVM-light package with the linear kernel and all default parameters. In the self-training process, we set the initial size of the negative set equal to that of the true positives labelled by the dictionary. In each iteration, we selected a fixed number (about 5% of total unlabelled tokens) of new negatives. We stopped the training iterations when 80% of unlabeled tokens had been labelled.

![Performance Comparison](image)

Figure 3. The overall F-scores of the prototype and baseline system under different dictionary coverages.

The overall F-scores of exact match on all the entity types are shown in Figure 3. From the results, we can see that our system beats the baseline system that uses the same dictionary, and approaches the skyline system when using larger dictionaries. For example, using the same 50% random subset of the perfect dictionary, our system achieves a 0.46 F-score, while the baseline system only has a 0.29 F-score.

We observe that even using the 100% dictionary to label the corpus, the performance of our system is still about 0.1 F-score less than the skyline performance. This may be due to the fact that we ignored the dictionary labels that contradict the true labels, and did not expand the positive set in the training process. As a result, the number of positive data is smaller than in the skyline system. Thus the trained model is less accurate. Also note that the improvement of the self-training upon the baseline was nearly constant (about 0.15 F-score) for every size of the used dictionary, which coincides with that of the skyline upon the baseline. These observations suggest that, given the learning ability of the base learner, properly expanding the positive set would be the key to further improving the final performance of the self-training method.

5 Conclusion and Future Work

We address the problem of how to recognize named entities from biomedical text in the absence of any human annotated corpus. The idea is to use a dictionary lookup to label the training sentences. Given the positive words labelled by the dictionary, we design an SVM-based self-training algorithm to identify negative words from unlabeled text so as to build an accurate classifier.

The purpose is to minimize the requirement of annotated corpora by traditional supervised learning algorithms. Our preliminary experiments suggest that this is possible, largely because the used dictionary is actually made by the domain experts. In this sense, the idea is somewhat equivalent to asking the dictionary to do the annotation job (as a dummy annotator).

We plan to study the following issues based on this work. First, whether the method can be applied to NER tasks in other domains (e.g., newswire), so as to develop the proposed technique into a general method. Although our intuition tells us that this is true, we still need to empirically demonstrate it. Second, we shall study how to improve the performance up to the level achieved by traditional supervised learning. Third, how to effectively identify reliable positive data from the unlabeled data. We believe the solution to the third issue will finally lead to solving the second issue.

Another interesting direction would be to apply our method in active learning for NER. In contrast to selecting the most reliable predictions as we were doing in this work, we could select the most unreliable ones for humans to label.
References


Hwanjo Yu, Jiawei Han, Kevin Chen-Chuan Chang. 2002. PEBL: positive example based learning for web page classification using SVM. In Proc. of KDD.
