ABSTRACT: Biomedical named entity recognition (Bio-NER) is the fundamental task of biomedical text mining. Machine-learning-based approaches, such as conditional random fields (CRFs), have been widely applied in this area, but the accuracy of these systems is limited because of the finite annotated corpus. In this study, word embedding features are generated from an unlabeled corpus, which as extra word features are induced into the CRFs system for Bio-NER. To further improved performance, a post-processing algorithm is employed after the named entity recognition task. Experimental results show that the word embedding features generated from a larger unlabeled corpus achieves higher performance, and the use of word embedding features increases F-measure on INLPBA04 data from 71.50% to 71.77%. After applying the post-processing algorithm, the F-measure reaches 71.85%, which is superior to the results in most existing systems.

Subject Categories and Descriptors
I.2.7 [Artificial Intelligence]: Natural Language Processing - Text Analysis; H.3.1 [Information Storage And Retrieval]: Content Analysis and Indexing - Linguistic processing

General Terms
Algorithm, Biomedical Text Mining

Keywords: Named Entity Recognition, Word Embeddings, Conditional Random Fields, Text Mining

Received: 9 June 2015, Revised 29 July 2015, Accepted 8 August 2015

1. Introduction

Biomedical named entity recognition (Bio-NER) is one of the basic tasks of biomedical text mining. The purpose of Bio-NER is to identify biomedical named entities (Bio-NEs), such as gene and protein. Owing to the complex nature of Bio-NEs, the recognition accuracy of the Bio-NER system is extremely lower compared to the recognition accuracy of standard entities, such as person names, location names, and government names in the newswire domain. Thus, Bio-NER remains a challenging task. First, a Bio-NE may have several written forms. For example, entity names (e.g., 5-lipoxygenase) often contain digits, alphabets, and other characters. Moreover, numerous abbreviations (e.g., “IL-2” for Interleukin-2) are used for Bio-NEs. Second, the same term can refer to more than one type of entity or not refer to an entity depending on the context (e.g., “T-cells” may refer to a cell_line or a cell_type). Third, many biomedical entities are phrases or compound words (e.g., “HIV enhancer element” is a DNA). All of these factors add to the difficulty of identifying Bio-NEs.

The main approaches for Bio-NER fall into three categories: dictionary-based approach, rule-based approach, and machine-learning-based approach. However, dictionary-based approaches are prone to miss undefined words that do not appear in the dictionary, while in rule-based approaches, the rules used to identify terms are critical, and building the rules demands much effort. The limitations of the two approaches above resulted in the widespread application of machine-learning-based approach in recent
years. Various machine learning algorithms have been introduced to solve the Bio-NER problem, such as hidden Markov model (HMM), support vector machine (SVM), and conditional random fields (CRFs). For example, Zhu et al. combined SVM-CRFs for Bio-NER, which achieved a macro-F1 of 91.67% on the GENIA corpus and of 84.04% on the INLPA 04 data [1]. Sun et al. developed a Bio-NER system based on a sequence memoizer which outperforms the HMM model and is comparable with the Maxent model [2]. Li et al. used CRFs to identify protein, DNA, RNA, cell_type, and cell_line with an F1 score of 74.31% [3].

A supervised machine learning algorithm uses annotated data for training and testing, and the success of a learning algorithm is dependent on the features it uses. Selecting the important feature values plays an important role in Bio-NER [4]. Bio-NER systems use different features, including orthographic, morphological, and local context features, some of which have been used by researchers and have achieved good performance. Sun et al. used a rich set of features, such as orthographical, word shape, context, and part-of-speech (POS) features, for Bio-NER and achieved an F-measure of 71.2% in open evaluation data [5]. Li et al. used word, POS, chunk, keyword, boundary word, prefix/suffix, and word shape features for Bio-NER and outperformed most of the state-of-the-art systems [3].

The main problem in supervised machine learning-based approaches is the lack of an annotated corpus, because of the costly human effort required to annotate text. The system performance is limited by the finite annotated corpus. The amount of unlabeled data in biomedical literature is increasing, which could provide extensive information for the identification task. Therefore, determining how unlabeled data can be utilized to improve performance is a problem that demands an urgent resolution [6].

Recently, using new features from unlabeled data to improve the accuracy of machine learning-based named entity recognition (NER) systems has increased. One technique that has gained popularity is word embedding, which induces a real-valued vector for each word from an unlabeled corpus. Word embedding features carry the latent syntactic and semantic information of a word, but very few studies have applied word embedding for Bio-NER [7].

The present study uses word embedding features as extra word features, and induces these features into a CRFs-based Bio-NER system. The effectiveness of our proposed method is proven by experiments.

### 2. Method

#### 2.1 Conditional Random Fields

CRFs are used to solve sequence labeling problems, and they show good performance in NER tasks. Linear-chain CRFs are often used in NER tasks. Given a word sequence \( w = \{w_i\}_{i=0}^n \) and its corresponding tag sequence \( t = \{t_i\}_{i=0}^n \), the conditional probability \( p(t \mid w) \) is computed as

\[
P(t \mid w) = \frac{1}{z} \exp \left( \sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_j f_j(t_{i-1}, t_i, w_i) \right)
\]

Where \( f_j(t_{i-1}, t_i, w_i) \) is the feature function, \( \lambda_j \) is the weight of feature, \( t_i \) and \( t_{i-1} \) are the current state and previous state respectively, and \( z \) is the normalization function:

\[
z = \sum_{t} \exp \left( \sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_j f_j(t_{i-1}, t_i, w_i) \right)
\]

For the application of a linear-chain CRF model, the key problem is how to find the parameters \( \theta \). For this purpose, we need the training data \( D = \{(w^{(i)}, t^{(i)})\}_{i=1}^n \) where each \( w^{(i)} = \{w_1^{(i)}, w_2^{(i)}, \ldots, w_n^{(i)}\} \) is an input sequence and each \( t^{(i)} = \{t_1^{(i)}, t_2^{(i)}, \ldots, t_n^{(i)}\} \) is a corresponding tag sequence. The objective of parameter learning is to find the weights that maximize the conditional log likelihood of the training data:

\[
\sum_{i=1}^{n} \log p(t^{(i)} \mid w^{(i)})
\]

A Gaussian prior on the \( \theta \) is involved to regularize the training and thus avoid overfitting. The formula (3) becomes

\[
\sum_{i=1}^{n} \log p(t^{(i)} \mid w^{(i)}) - \frac{1}{2\sigma^2} \sum_{j=1}^{m} \lambda_j^2
\]

where \( \sigma^2 \) is the strength of the penalty [5].

The Bio-NER task is considered as a sequence labeling problem where each token in a sequence is assigned a biological name label. In this study, the corpus is annotated utilizing the BIO format, where “B-ne” denotes the words that are the beginning of an entity, “I-ne” refers to the rest of the entity, and “O” indicates a token that is not part of an entity. An example of Bio-NER is listed in Table 1.

The feature function \( f(t_{i-1}, t_i, w_i) \) for the Bio-NER problem is a binary-valued function, which may be defined as

\[
f(t_{i-1}, t_i, w_i) = \begin{cases} 1 & \text{if } w_i = \text{gene}, t_{i-1} = \text{expression}, \text{t} \neq \text{B-NA}, \text{t} = \text{I-NA} \\ 0 & \text{otherwise} \end{cases}
\]

Table 1. Example of named entities represented by BIO labels

<table>
<thead>
<tr>
<th>Token</th>
<th>IL-2</th>
<th>Gene</th>
<th>Expression</th>
<th>And</th>
<th>NF-kappa</th>
<th>B</th>
<th>Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>B-DNA</td>
<td>I-DNA</td>
<td>O</td>
<td>O</td>
<td>B-protein</td>
<td>I-protein</td>
<td>O</td>
</tr>
</tbody>
</table>
The labeling for a new unlabeled sequence can be performed using a modified Viterbi algorithm. A CRF++ (version 0.58) toolkit is used to implement the CRFs model in our system.

2.2 Features
The performance of a CRFs model depends on the features. To evaluate the system performance, the features used to develop the system are shown below.

2.2.1 General Features
General features include word, POS, chunk, orthographic prefix/suffix, and word shape features. These features are easy to derive and do not require biomedical domain knowledge.

a. Word features \((F_w)\). We employ a word window of size 5, which consists of the current word, the two previous words, and the two following words. Features in the form of unigrams, bigrams, and trigrams are also included.

b. POS features \((F_{POS})\). GENIA Tagger [8] is used to derive POS features; it is trained on biological literature and has an accuracy of up to 98.20%. POS features can help to improve the generalization of the model, and also include unigram, bigram, and trigram types.

c. Chunk features \((F_c)\). Chunk features can help to recognize Bio-NE boundaries. GENIA Tagger is also used to obtain chunk features for each token and the same window size as POS features.

d. Orthographic features \((F_o)\). Orthographic features are used to capture word formation information, such as digit character and single character. These features group words by similar formations. For example, IL-2 is similar to IL-4 on the basis of their orthographic features. Table 2 lists the orthographic features used in this paper.

e. Prefix/suffix features \((F_{fix})\). Prefixes and suffixes are indicative of the existence of Bio-NE. For example, tokens ending in “ase” are usually proteins. All prefixes and suffixes with a length of three to five were used in our experiment.

f. Word shape features \((F_s)\). Tokens with similar word shapes may belong to the same class. All upper-case characters are replaced by “X,” all lower-case characters are replaced by “x,” digits are replaced by “0,” and other characters are replaced by “.”. For example, “IL-2” is normalized as “XX_0.”

2.2.2 Word Embeddings
Distributed representation is a type of word representation. Distributed word representations are also called word embeddings, which induce a real-valued vector for each word from an unlabeled corpus through continuous space language models. Each dimension of the embedding represents a latent feature of the word, capturing useful semantic and syntactic information. Word embeddings, when used as the underlying input representation, boosts the performance in NLP tasks, such as syntactic parsing and sentiment analysis.

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Regular expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLCAPS</td>
<td>[A-Z]+</td>
</tr>
<tr>
<td>INITCAP</td>
<td>^[A-Z].*</td>
</tr>
<tr>
<td>CAPSMIX</td>
<td>.<em>[A-Z][a-z].</em></td>
</tr>
<tr>
<td>SINGLE CHAR</td>
<td>[A-Za-z]</td>
</tr>
<tr>
<td>HAS DIGIT</td>
<td>.<em>[0-9].</em></td>
</tr>
<tr>
<td>SINGLE DIGIT</td>
<td>[0-9]</td>
</tr>
<tr>
<td>DOUBLE DIGIT</td>
<td>[0-9][0-9]</td>
</tr>
<tr>
<td>NATURAL NUMBER</td>
<td>[0-9]+</td>
</tr>
<tr>
<td>REAL NUMBER</td>
<td>[-0-9]+[,]+[0-9,]+</td>
</tr>
<tr>
<td>HAS DASH</td>
<td>.*_.</td>
</tr>
<tr>
<td>INIT DASH</td>
<td>_.</td>
</tr>
<tr>
<td>END DASH</td>
<td>_.</td>
</tr>
<tr>
<td>ALPHA NUMERIC</td>
<td>([A-Za-z][0-9].*)</td>
</tr>
<tr>
<td>ROMAN</td>
<td>[IVXDLCM]+</td>
</tr>
<tr>
<td>PUNCTUATION</td>
<td>[.,!?+:]+</td>
</tr>
</tbody>
</table>

Table 2. Orthographic features

Various models for learning word embeddings have been proposed, such as neural net language models and spectral models. Mikolov et al. [10] proposed two log-linear models, namely, skip-gram and CBOW, to induce word embeddings. These two models can be trained efficiently on a large amount of text data owing to their low time complexity. The skip-gram model performs better than the CBOW model in identifying semantic relationship among words, and is used to estimate word embeddings in this study.

The skip-gram model adopts log-linear classifiers to predict context words. Given a sequence of training words \(w_1, w_2, \ldots, w_n\), the objective of the skip-gram model is to maximize the average log probability:

\[
\frac{1}{n} \sum_{t=1}^{n} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)
\]

where \(c\) is the size of the training context, and \(p(w_{t+j} | w_t)\) is defined through the softmax function:

\[
p(w_{t+j} | w_t) = \frac{\exp(v_{w_{t+j}}^T v_{w_t})}{\sum_{w} \exp(v_{w}^T v_{w_t})}
\]

where \(v_{w_t}\) is the input vector representations of \(w_t\), \(v_{w_{t+j}}\) is the output vector representations of \(w_t\), and \(W\) is the number of words in the vocabulary [9–12].

In the experiment, we utilize the “word2vec” deep learning...
toolkit (https://code.google.com/p/word2vec/) to generate word embeddings, and the dimensionality of the word vectors is 100 (as shown in Table 3).

Table 3 Example of word embedding features

| IL-2:          | -0.004383 -0.002921 0.001765 0.003527 0.000247 0.001568 0.003372 0.000324 0.001129 0.003362 -0.002705 0.000138 -0.004085 -0.002976 …… |
| gene:         | 0.003242 0.001329 0.001803 -0.002194 0.004607 -0.004003 0.002604 0.003039 0.003314 -0.004202 -0.002769 0.001647 0.003988 0.000240 …… |
| expression:   | 0.002291 0.001110 0.000353 0.002255 -0.004162 -0.001495 -0.000814 0.004616 0.003458 0.004780 0.000619 0.004062 -0.002204 …… |
| and:          | -0.004739 0.002767 0.001354 0.002818 0.003671 0.000282 0.000831 -0.004563 -0.001495 0.001784 0.004196 -0.001213 -0.003879 …… |
| NF-kappa:     | 0.003260 0.001533 0.003310 0.004572 0.000341 -0.002144 0.004855 0.00165 -0.003494 0.000874 0.000783 0.001146 0.000453 -0.004876 …… |
| B:            | 0.002653 -0.002301 -0.002111 -0.004088 0.001899 0.002485 0.001446 0.003452 -0.004186 0.004264 0.003208 0.001231 -0.002964 0.002601 …… |

2.3 Post-processing Algorithms

If two biomedical entities are connected by “and” or “or”, and they modify the same noun, then they can be merged into one. Table 4 presents an example.

<table>
<thead>
<tr>
<th>Before merge</th>
<th>After merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL-4 /NN</td>
<td>B</td>
</tr>
<tr>
<td>and/CC</td>
<td>O</td>
</tr>
<tr>
<td>IL-10 /NN</td>
<td>B</td>
</tr>
<tr>
<td>transcripts</td>
<td>/NNS</td>
</tr>
</tbody>
</table>

Table 4. Example of merging entities connected by “and” or “or”

In Table 4, “transcripts” was modified by both “IL-4” and “IL-10,” so “IL-4” and “IL-10 transcripts” can be merged as “IL-4 and IL-10 transcripts.” The similar words also include “gene,” “kinases,” and “antibodies.”

3. Experiments and Analysis

In this section, we discuss the experiment data sets and report the experiment results.

3.1 Experiment Data Sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>#protein</th>
<th>#DNA</th>
<th>#RNA</th>
<th>#cell_type</th>
<th>#cell_line</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>30269</td>
<td>9533</td>
<td>951</td>
<td>6718</td>
<td>3830</td>
<td>51031</td>
</tr>
<tr>
<td>Test set</td>
<td>5067</td>
<td>1056</td>
<td>118</td>
<td>1921</td>
<td>500</td>
<td>8662</td>
</tr>
</tbody>
</table>

Table 5. Entity distribution in the JNLPBA data set

3.2 Experiment Results

The JNLPBA training set and test set were used to train and evaluate the model respectively. Precision, recall and F1-measure were used to evaluate the effectiveness of our method. Precision, recall, and F-measure are defined as follows:

\[
\text{Precision} = \frac{\# \text{ of correctly recognized named entities}}{\text{Total \# of named entities recognized}} \quad (8)
\]

\[
\text{Recall} = \frac{\# \text{ of correctly recognized named entities}}{\text{Total \# of named entities identified by a domain expert}} \quad (9)
\]

\[
F - \text{measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (10)
\]

3.2.1 Effectiveness of Different Features

Table 6 summarizes the results when different features were used.

Several findings can be concluded from Table 6:

(1) Based on word features, the system can achieve an F-measure of 66.67%, which proves that word features are very important for Bio-NER.
(3) Chunk features are somewhat useful because their use only increases the F-measure by 0.08% based on $F_w + F_{POS}$. Therefore, chunk features were not used in later experiments.

(4) Introducing orthographic features can increase the F-measure by 0.82% and recall by 3.14% based on $F_w + F_{POS}$. The reason for this effect could be the ability of orthographic features to help detect entities that do not appear in the training dataset.

(5) The use of prefix/suffix features improved the F-measure by 1.06% based on $F_w + F_{POS} + F_{c}$. Prefixes and suffixes can provide good clues for biomedical entities. Additional experiments on increasing the length of the prefix/suffix may further improve the performance of the model.

(6) Word shape features are not as useful as expected. The application of word shape features increases F-measure by 0.07% based on $F_w + F_{POS} + F_o + F_{fix}$. Therefore, word shape features were not used in later experiments.

(7) Word embedding features are somewhat useful because they only increase F-measure by 0.18% based on $F_w + F_{POS} + F_o + F_{fix}$. One reason for this result may be that the majority of information has been covered by word features. Another reason may be that the unlabeled corpus used to generate word embedding features is not large enough.

To verify our inference, further experiments were conducted on the basis of two different word embedding features. One was generated from the JNLPBA04 corpus, and the other was derived from the unlabeled corpora of JNLPBA04, BioNLP09, and BioNLP11.

We started with a system that had basic features, including word, POS, orthographic, and prefix/suffix features, and then evaluated the effect of two word embedding features. The results are shown in Figure 1. Figure 1 illustrates that both word embedding features can increase system performance; when the unlabeled corpus is increased, the system performance increases from 71.68% to 71.77%.

The detailed results are listed in Table 7. In Table 7, We concluded that the word embedding features generated from a larger unlabeled corpus achieves higher performance.

<table>
<thead>
<tr>
<th>$F_w$</th>
<th>$F_{POS}$</th>
<th>$F_c$</th>
<th>$F_o$</th>
<th>$F_{fix}$</th>
<th>$F_{s}$</th>
<th>$F_e$</th>
<th>P (%)</th>
<th>R (%)</th>
<th>F (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>70.73</td>
<td>63.06</td>
<td>66.67</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td>71.25</td>
<td>68.06</td>
<td>69.62</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td>71.38</td>
<td>68.09</td>
<td>69.70</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td>69.70</td>
<td>71.20</td>
<td>70.44</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td>71.15</td>
<td>71.84</td>
<td>71.50</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td>71.31</td>
<td>71.84</td>
<td>71.57</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td></td>
<td>71.50</td>
<td>72.06</td>
<td>71.68</td>
</tr>
</tbody>
</table>

Table 6. Experimental results using the features mentioned in Section 2.2

$F_w$ refers to word features, $F_{POS}$ refers to POS features, $F_c$ refers to chunk features, $F_o$ refers to orthographic features, $F_{fix}$ refers to prefix/suffix features, $F_s$ refers to word shape features, and $F_e$ refers to word embedding features.

Figure 1. $F_{e1}$ denotes word embedding features generated from the unlabeled corpus of JNLPBA04, $F_{e2}$ denotes word embedding features generated from the unlabeled corpora of JNLPBA04, BioNLP09, and BioNLP11.
3.2.2 Comparison with Existing Bio-NER Systems

Our best results (an F-measure of 71.77%, as shown in Table 7) are post-processed by algorithms described in Section 2.3, and we achieve an F-measure of 71.85% (The results are shown in Table 8).

<table>
<thead>
<tr>
<th>System</th>
<th>ML. approach</th>
<th>Domain knowledge</th>
<th>F (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ponomareva et al. (2007)</td>
<td>HMM</td>
<td>POS</td>
<td>65.7</td>
</tr>
<tr>
<td>Saha et al. (2009)</td>
<td>MaxEnt</td>
<td>POS</td>
<td>67.41</td>
</tr>
<tr>
<td>Sun et al. (2007)</td>
<td>CRFs</td>
<td>POS</td>
<td>71.2</td>
</tr>
<tr>
<td>Our system</td>
<td>CRFs</td>
<td>POS, word embedding</td>
<td>71.85</td>
</tr>
</tbody>
</table>

Table 8. Results of comparison with other systems

In Table 8, Ponamareva et al. [14] developed an HMM-based system and achieved an F-measure value of 65.7%. Saha et al. [4] developed a MaxEnt based system employing reduced features and the F-measure value is 67.41%. Sun et al. [5] used a rich set of features for the CRFs-based system and achieved an F-measure value of 71.2%. Our system introduced word embedding features and achieved the highest F-measure value of 71.85%. Table 8 shows that our system is superior to other systems.

4. Conclusion

In this study, we utilize a CRFs-based machine learning approach for Bio-NER. The proposed system uses different features, some of which are language independent and can be applied for other languages. The system also employs word embedding features that induce a real-valued vector for each word from the unlabeled corpus. Experiment results show that the accuracy of the system can be further improved by inducing word embedding features, and the system has better performance than other existing systems that do not use deep domain knowledge.

Acknowledgement

This work is supported by 111 Project of China under Grant No.B08004, key project of ministry of science and technology of China under Grant No. 2011ZX03002-005-01, National Natural Science Foundation of China (61273217) and the Ph.D. Programs Foundation of Ministry of Education of China (20130005110004).

References


