Boosting Precision and Recall of Dictionary-Based Protein Name Recognition

Yoshimasa Tsuruoka‡† and Jun'ichi Tsujii†‡
†Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 Japan
‡CREST, JST (Japan Science and Technology Corporation)
Honcho 4-1-8, Kawaguchi-shi, Saitama 332-0012 Japan
{tsuruoka,tsujii}@is.s.u-tokyo.ac.jp

Abstract

Dictionary-based protein name recognition is the first step for practical information extraction from biomedical documents because it provides ID information of recognized terms unlike machine learning based approaches. However, dictionary based approaches have two serious problems: (1) a large number of false recognitions mainly caused by short names. (2) low recall due to spelling variation. In this paper, we tackle the former problem by using a machine learning method to filter out false positives. We also present an approximate string searching method to alleviate the latter problem. Experimental results using the GENIA corpus show that the filtering using a naive Bayes classifier greatly improves precision with slight loss of recall, resulting in a much better F-score.

1 Introduction

The rapid increase of machine readable biomedical texts (e.g. MEDLINE) makes automatic information extraction from those texts much more attractive. Especially extracting information of protein-protein interactions from MEDLINE abstracts is regarded as one of the most important tasks today (Marcotte et al., 2001; Thomas et al., 2000; Ono et al., 2001).

To extract information of proteins, one has to first recognize protein names in a text. This kind of problem has been studied in the field of natural language processing as named entity recognition tasks. Ohta et al. (2002) provided the GENIA corpus, an annotated corpus of MEDLINE abstracts, which can be used as a gold-standard for evaluating and training named entity recognition algorithms. There are some research efforts using machine learning techniques to recognize biological entities in texts (Takeuchi and Collier, 2002; Kim and Tsujii, 2002; Kazama et al., 2002).

One drawback of these machine learning based approaches is that they do not provide identification information of recognized terms. For the purpose of information extraction of protein-protein interaction, the ID information of recognized proteins, such as GenBank [1] ID or SwissProt [2] ID, is indispensable to integrate the extracted information with the data in other information sources.

Dictionary-based approaches, on the other hand, intrinsically provide ID information because they recognize a term by searching the most similar (or identical) one in the dictionary to the target term. This advantage currently makes dictionary-based approaches particularly useful as the first step for practical information extraction from biomedical documents (Ono et al., 2001).

However, dictionary-based approaches have two serious problems. One is a large number of false positives mainly caused by short names, which significantly degrade overall precision. Although this problem can be avoided by excluding short names from the dictionary, such a solution makes it impossible to recognize short protein names. We tackle

[1] GenBank is one of the largest genetic sequence databases.
[2] The Swiss-Prot is an annotated protein sequence database.

this problem by using a machine learning technique. Each recognized candidate is checked if it is really protein name or not by a classifier trained on an annotated corpus.

The other problem of dictionary based approaches is spelling variation. For example, the protein name “NF-Kappa B” has many spelling variants such as “NF Kappa B,” “NF kappa B,” “NF kappaB,” and “NFkappaB.” Exact matching techniques, however, regard these terms as completely different terms. We alleviate this problem by using an approximate string matching method in which surface-level similarities between terms are considered.

This paper is organized as follows. Section 2 describes the overview of our method. Section 3 presents the approximate string searching algorithm for candidate recognition. Section 3 describes how to filter out false recognitions by a machine learning method. Section 5 presents the experimental results using the GENIA corpus. Some related work is described in Section 6. Finally, Section 7 offers some concluding remarks.

2 Method Overview

Our protein name recognition method consists of two phases. In the first phase, we scan the text for protein name candidates using a dictionary. In the second phase, we check each candidate whether it is really protein name or not using a machine learning method. We call these two phases *recognition phase* and *filtering phase* respectively. The overview of the method is given below.

- Recognition phase

  Protein name candidates are identified using a protein name dictionary. To alleviate the problem of spelling variation, we use an approximate string matching technique.

- Filtering phase

  Every protein name candidates is classified into “accepted” or “rejected” by a classifier. The classifier uses the context of the term and the term itself as the features for the classification. Only “accepted” candidates are recognized as protein names.

In the following sections, we describe the details of each phase.

		G	R	-	2
	0	1	2	3	4
E	1	1	2	3	4
G	2	1	2	3	4
R	3	2	1	2	3
-	4	3	2	1	2
1	5	4	3	2	2

Figure 1: Dynamic Programming Matrix

3 Candidate Recognition

The most straightforward way to exploit a dictionary for candidate recognition is the exact (longest) match algorithm. For exact match, many fast matching algorithms (e.g. Boyer-Moore algorithm (1977)) have been proposed. However, the existence of many spelling variations for the same protein name makes the exact matching less attractive. For example, even a short protein name “EGR-1” has at least the six following variations:

EGR-1, EGR 1, Egr-1, Egr 1, egr-1, egr 1.

Since longer protein names have a huge number of possible variations, it is impossible to enrich the dictionary by expanding each protein name as described above.

3.1 Approximate String Searching

To deal with the problem of spelling variation, we need a kind of ‘elastic’ matching algorithm, by which a recognition system scan a text to find a similar term to (if any) a protein name in the dictionary. We need a similarity measure to do such a task. The most popular measure of similarity between two strings is *edit distance*, which is the minimum number of operations on individual characters (e.g. substitutions, insertions, and deletions) required to transform one string of symbols into another. For example, the edit distance between “EGR-1” and “GR-2” is two, because one substitution (1 for 2) and one deletion (E) are required.

To calculate the edit distance between two strings, we can use a dynamic programming technique. Figure 1 illustrates an example. For clarity of presentation, all costs are assumed to be 1. The matrix $C_{0..|x|,0..|y|}$ is filled, where $C_{i,j}$ represents the minimum number of operations needed to match $x_{1..i}$ to $y_{1..j}$. This is computed as follows (Navarro, 1998)

$$
\begin{aligned}
C_{i,0} &= i && (1)\\
C_{0,j} &= j && (2)\\
C_{i,j} &= \text{if } (x_i = y_j) \text{ then } C_{i-1,j-1} && (3)\\
&\quad \text{else } 1 + \min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1})
\end{aligned}
$$

The calculation can be done by either a row-wise left-to-right traversal or a column-wise top-to-bottom traversal.

There are many fast algorithms other than the dynamic programming for uniform-cost edit distance, where the weight of each edit operation is constant within the same type (Navarro, 2001). However, what we expect is that the distance between "EGR-1" and "EGR 1" will be smaller than that between "EGR-1" and "FGR-1", while the uniform-cost edit distances of them are equal.

The dynamic programming based method is flexible enough to allow us to define arbitrary costs for individual operations depending on a letter being operated. For example, we can make the cost of the substitution between a space and a hyphen much lower than that of the substitution between 'E' and 'F.' Therefore, we use the dynamic programming based method for our task.

Table 1 shows the cost function used in our experiments. Both insertion and deletion costs are 100 except for spaces and hyphens. Substitution costs for similar letters are 10. Substitution costs for the other different letters are 50.

Table 1: Cost Function

Operation	Letter	Cost
Insertion	*space* or *hyphen*	10
	Other letters	100
Deletion	*space* or *hyphen*	10
	Other letters	100
Substitution	A letter for the same letter	0
	A numeral for a numeral	10
	space for *hyphen*	10
	hyphen for *space*	10
	A capital letter for the corresponding small letter	10
	A small letter for the corresponding capital letter	10
	Other letters	50

3.2 String Searching

We have described a method for calculating the similarity between two strings in the previous section. However, what we need is approximate string searching in which the recognizer scans a text to find a similar term to (if any) a term in the dictionary. The dynamic programming based method can be easily extended for approximate string searching.

The method is illustrated in Figure 2. The protein name to be matched is "EGR-1" and the text to be scanned is "encoded by EGR include." String searching can be done by just setting the elements corresponding separators (e.g. space) in the first row to zero. After filling the whole matrix, one can find that "EGR-1" can be matched to this text at the place of "EGR 1" with cost 1 by searching for the lowest value in the bottom row.

To take into account the length of a term, we adopt a normalized cost, which is calculated by dividing the cost by the length of the term:

$$
\text{(nomalized cost)} = \frac{\text{(cost)} + \alpha}{\text{(length of the term)}} \quad (4)
$$

where α is a constant value [3]. When the costs of two terms are the same, the longer one is preferred due to this constant.

To recognize a protein name in a given text, we perform the above calculation for every term contained in the dictionary and select the term that has the lowest normalized cost.

If the normalized cost is lower than the predefined threshold. The corresponding range in the text is recognized as a protein name candidate.

3.3 Implementation Issues for String Searching

A naive way for string searching using a dictionary is to conduct the procedure described in the previous section one by one for every term in the dictionary. However, since the size of the dictionary is very large, this naive method takes too much time to perform a large scale experiment.

[3] α was set to 0.4 in our experiments.

		e	n	c	o	d	e	d		b	y		E	G	R		1		i	n	c	l	u	d	e
	0	1	2	3	4	5	6	7	0	1	2	0	1	2	3	0	1	0	1	2	3	4	5	6	7
E	1	1	2	3	4	5	6	7	1	1	2	1	0	1	2	1	1	1	1	2	3	4	5	6	7
G	2	2	2	3	4	5	6	7	2	2	2	2	1	0	1	2	2	2	2	2	3	4	5	6	7
R	3	3	3	3	4	5	6	7	3	3	3	3	2	1	0	1	2	3	3	3	3	4	5	6	7
-	4	4	4	4	4	5	6	7	4	4	4	4	3	2	1	1	2	3	4	4	4	4	5	6	7
1	5	5	5	5	5	5	6	7	5	5	5	5	4	3	2	2	1	2	3	4	5	5	5	6	7

Figure 2: Example of String Searching using Dynamic Programming Matrix

Navarro (2001) have presented a way to reduce redundant calculations by constructing a trie of the dictionary. The trie is used as a device to avoid repeating the computation of the cost against same prefix of many patterns. Suppose that we have just calculated the cost of the term "EGR-1" and next we have to calculate the cost of the term "EGR-2," it is clear that we do not have to re-calculated the first four rows in the matrix (see Figure 2). They also pointed out that it is possible to determine, prior to reaching the bottom of the matrix, that the current term cannot produce any relevant match: if all the values of the current row are larger than the threshold, then a match cannot occur since we can only increase the cost or at best keep it the same.

4 Filtering Candidates by a Naive Bayes Classifier

One of the serious problems of dictionary-based recognition is a large number of false recognitions mainly caused by short entries in the dictionary. For example, the dictionary constructed from GenBank contains an entry "NK." However, the word "NK" is frequently used as a part of the term "NK cells." In this case, "NK" is an abbreviation of "natural killer" and is not a protein name. Therefore this entry makes a large number of false recognitions leading to low precision performance.

In the filtering phase, we use a classifier trained on an annotated corpus to suppress such kind of false recognition. The objective of this phase is to improve precision without the loss of recall.

We conduct binary classification ("accept" or "reject") on each candidate. The candidates that are classified into "rejected" are filtered out. In other words, only the candidates that are classified into "accepted" are recognized as protein names.

In this paper, we use a naive Bayes classifier for this classification task.

4.1 Naive Bayes classifier

The naive Bayes classifier is a simple but effective classifier which has been used in numerous applications of information processing such as image recognition, natural language processing and information retrieval (Lewis, 1998; Escudero et al., 2000; Pedersen, 2000; Nigam and Ghani, 2000).

Here we briefly review the naive Bayes model. Let $\vec{x}$ be a vector we want to classify, and c_k be a possible class. What we want to know is the probability that the vector $\vec{x}$ belongs to the class c_k. We first transform the probability $P(c_k|\vec{x})$ using Bayes' rule,

$$P(c_k|\vec{x}) = P(c_k) \times \frac{P(\vec{x}|c_k)}{P(\vec{x})} \qquad (5)$$

Class probability $P(c_k)$ can be estimated from training data. However, direct estimation of $P(c_k|\vec{x})$ is impossible in most cases because of the sparseness of training data.

By assuming the conditional independence among the elements of a vector, $P(\vec{x}|c_k)$ is decomposed as follows,

$$P(\vec{x}|c_k) = \prod_{j=1}^{d} P(x_j|c_k), \qquad (6)$$

where x_j is the jth element of vector $\vec{x}$. Then Equation 5 becomes

$$P(c_k|\vec{x}) = P(c_k) \times \frac{\prod_{j=1}^{d} P(x_j|c_k)}{P(\vec{x})} \quad (7)$$

By this equation, we can calculate $P(c_k|\vec{x})$ and classify $\vec{x}$ into the class with the highest $P(c_k|\vec{x})$.

There are some implementation variants of the naive Bayes classifier depending on their event models (McCallum and Nigam, 1998). In this paper, we adopt the multi-variate Bernoulli event model.

4.2 Features

As the input of the classifier, the features of the target must be represented in the form of a vector. We use a binary feature vector which contains only the values of 0 or 1 for each element.

In this paper, we use the local context surrounding a candidate term and the words contained in the term as the features. We call the former *contextual features* and the latter *term features*.

The features used in our experiments are given below.

- Contextual Features

  W_{-1} : the preceding word.

  W_{+1} : the following word.

- Term Features

  W_{begin} : the first word of the term.

  W_{end} : the last word of the term.

  W_{middle} : the other words of the term without positional information (bag-of-words).

Suppose the candidate term is "putative zinc finger protein, " and the sentence is:

... encoding a putative zinc finger protein was found to derepress beta- galactosidase ...

We obtain the following active features for this example.

$\{W_{-1}$ a$\}$, $\{W_{+1}$ was$\}$, $\{W_{begin}$ putative$\}$, $\{W_{end}$ protein$\}$, $\{W_{middle}$ zinc$\}$, $\{W_{middle}$ finger$\}$.

4.3 Training

The training of the classifier is done with an annotated corpus. We first scan the corpus for protein name candidates by dictionary matching. If a recognized candidate is annotated as a protein name, this candidate and its context are used as a positive ("accepted") example for training. Otherwise, it is used as a negative ("rejected") example.

5 Experiment

5.1 Corpus and Dictionary

We conducted experiments of protein name recognition using the GENIA corpus version 3.01 (Ohta et al., 2002). The GENIA corpus is an annotated corpus, which contains 2000 abstracts extracted from MEDLINE database. These abstracts are selected from the search results with MeSH terms *Human*, *Blood Cells*, and *Transcription Factors*.

The biological entities in the corpus are annotated according to the GENIA ontology. Although the corpus has many categories such as protein, DNA, RNA, cell line and tissue, we used only the protein category. When a term was recursively annotated, only the innermost (shortest) annotation was considered.

The test data was created by randomly selecting 200 abstracts from the corpus. The remaining 1800 abstracts were used as the training data. The protein name dictionary was constructed from the training data by gathering all the terms that were annotated as proteins.

Each recognition was counted as correct if the both boundaries of the recognized term exactly matched the boundaries of an annotation in the corpus.

5.2 Improving Precision by Filtering

We first conducted experiments to evaluate how much precision is improved by the filtering process. In the recognition phase, the longest matching algorithm was used for candidate recognition.

The results are shown in Table 2. F-measure is defined as the harmonic mean for precision and recall as follows:

$$F = \frac{2 \times precision \times recall}{precision + recall} \quad (8)$$

Table 2: Precision Improvement by Filtering

	Precision	Recall	F-measure
w/o filtering	48.6	70.7	57.6
with filtering	74.3	65.3	69.5

Table 3: Recall Improvement by Approximate String Search

Threshold	Precision	Recall	F-measure
1.0	72.6	39.5	51.2
2.0	73.7	63.7	68.3
3.0	74.0	66.5	70.1
4.0	73.9	66.8	70.2
5.0	73.4	67.1	70.1
6.0	73.6	67.1	70.2
7.0	73.5	67.2	70.2
8.0	73.1	67.4	70.2
9.0	72.9	67.8	70.2
10.0	72.6	67.7	70.0

The first row shows the performances achieved without filtering. In this case, all the candidates identified in the recognition phase are regarded as protein names. The second row shows the performance achieved with filtering by the naive Bayes classifier. In this case, only the candidates that are classified into “accepted” are regarded as protein names. Notice that the filtering significantly improved the precision (from 48.6% to 74.3%) with slight loss of the recall. The F-measure was also greatly improved (from 57.6% to 69.5%).

5.3 Improving Recall by Approximate String Search

We also conducted experiments to evaluate how much we can further improve the recognition performance by using the approximate string searching method described in Section 3. Table 3 shows the results. The leftmost columns show the thresholds of the normalized costs for approximate string searching. As the threshold increased, the precision degraded while the recall improved. The best F-measure was 70.2%, which is better than that of exact matching by 0.7% (see Table 2).

Table 4: Performance using Different Feature Set

Feature Set	Precision	Recall	F-measure
Contextual features	61.0	62.6	61.8
Term features	71.3	67.9	69.5
All features	73.5	67.2	70.2

5.4 Efficacy of Contextual Features

The advantage of using a machine learning technique is that we can exploit the context of a candidate for deciding whether it is really protein name or not. In order to evaluate the efficacy of contexts, we conducted experiments using different feature sets. The threshold of normalized cost was set to 7.0.

Table 4 shows the results. The first row shows the performances achieved by using only contextual features. The second row shows those achieved by using only term features. The performances achieved by using both feature sets are shown in the third row.

The results indicate that candidate terms themselves are strong cues for classification. However, the fact that the best performance was achieved when both feature sets were used suggests that the context of a candidate conveys useful information about the semantic class of the candidate.

6 Related Work

Kazama et al. (2002) reported an F-measure of 56.5% on the GENIA corpus (Version 1.1) using Support Vector Machines. Collier et al. (2001) reported an F-measure of 75.9% evaluated on 100 MEDLINE abstracts using a Hidden Markov Model. These research efforts are machine learning based and do not provide ID information of recognized terms.

Krauthammer et al. (2000) proposed a dictionary-based gene/protein name recognition method. They used BLAST for approximate string matching by mapping sequences of text characters into sequences of nucleotides that can be processed by BLAST. They achieved a recall of 78.8% and a precision of 71.1% by a partial match criterion, which is less strict than our exact match criterion.

7 Conclusion

In this paper we propose a two-phase protein name recognition method. In the first phase, we scan texts for protein name candidates using a protein name dictionary and an approximate string searching technique. In the second phase, we filter the candidates using a machine learning technique.

Since our method is dictionary-based, it can provide ID information of recognized terms unlike machine learning based approaches. False recognition, which is a common problem of dictionary-based approaches, is suppressed by a classifier trained on an annotated corpus.

Experimental results using the GENIA corpus show that the filtering using a naive Bayes classifier greatly improves precision with slight loss of recall. We achieved an F-measure of 70.2% for protein name recognition on the GENIA corpus.

The future direction of this research involves:

- Use of state-of-the-art classifiers

  We have used a naive Bayes classifier in our experiments because it requires a small computational resource and exhibits good performance. There is a chance, however, to improve performance by using state-of-the-art machine learning techniques including maximum entropy models and support vector machines.

- Use of other elastic matching algorithms

  We have restricted the computation of similarity to edit distance. However, it is not uncommon that the order of the words in a protein name is altered, for example,

  "beta-1 integrin"

  "integrin beta-1"

  The character-level edit distance cannot capture this -kind of similarities.

References

Robert S. Boyer and J. Strother Moore. 1977. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772.

Nigel Collier, Chikashi Nobata, and Junichi Tsujii. 2001. Automatic acquisition and classification of molecular biology terminology using a tagged corpus. *Journal of Terminology*, 7(2):239–258.

G. Escudero, L. arquez, and G. Rigau. 2000. Naive bayes and exemplar-based approaches to word sense disambiguation revisited. In *Proceedings of the 14th European Conference on Artificial Intelligence*.

Jun'ichi Kazama, Takaki Makino, Yoshihiro Ohta, and Jun'ichi Tsujii. 2002. Tuning support vector machines for biomedical named entity recognition. In *Proceedings of the ACL-02 Workshop on Natural Language Processing in the Biomedical Domain*.

Jin Dong Kim and Jun'ichi Tsujii. 2002. Corpus-based approach to biological entity recognition. In *Text Data Mining SIG (ISMB2002)*.

Michael Krauthammer, Andrey Rzhetsky, Pavel Morozov, and Carol Friedman. 2000. Using BLAST for identifying gene and protein names in journal articles. *Gene*, 259:245–252.

David D. Lewis. 1998. Naive Bayes at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15.

Edward M. Marcotte, Ioannis Xenarios, and David Eisenberg. 2001. Mining literature for protein-protein interactions. *BIOINFORMATICS*, 17(4):359–363.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. *In AAAI-98 Workshop on Learning for Text Categorization*.

G. Navarro, R. Baeza-Yates, and J.M. Arcoverde. 2001. Matchsimile: A flexible approximate matching tool for personal names searching. In *Proceedings of the XVI Brazilian Symposium on Databases (SBBD'2001)*, pages 228–242.

Gonzalo Navarro. 1998. *Approximate Text Searching*. Ph.D. thesis, Dept. of Computer Science, Univ. of Chile.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *CIKM*, pages 86–93.

Tomoko Ohta, Yuka Tateishi, Hideki Mima, and Jun'ichi Tsujii. 2002. Genia corpus: an annotated research abstract corpus in molecular biology domain. In *Proceedings of the Human Language Technology Conference*.

Toshihide Ono, Haretsugu Hishigaki, Akira Tanigami, and Toshihisa Takagi. 2001. Automated extraction of information on protein-protein interactions from the biological literature. *BIOINFORMATICS*, 17(2):155–161.

Ted Pedersen. 2000. A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 63–69.

K. Takeuchi and N. Collier. 2002. Use of support vector machines in extended named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, pages 119–125.

James Thomas, David Milward, Christos Ouzounis, Stephen Pulman, and Mark Carroll. 2000. Automatic extraction of protein interactions from scientific abstracts. In *Proceedings of the Pacific Symposium on Biocomputing (PSB2000)*, volume 5, pages 502–513.