Protein Name Tagging for Biomedical Annotation in Text

Kaoru Yamamoto† **Taku Kudo**‡ **Akihiko Konagaya**† **Yuji Matsumoto**‡
†Genomic Sciences Center, The Institute of Physical and Chemical Research
1-7-22-E209, Suehiro-cho, Tsurumi-ku, Yokohama, 230-0045 Japan
kaorux@gsc.riken.go.jp, konagaya@gsc.riken.go.jp
‡Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192 Japan
taku-ku@is.aist-nara.ac.jp, matsu@is.aist-nara.ac.jp

Abstract

We explore the use of morphological analysis as preprocessing for protein name tagging. Our method finds protein names by chunking based on a *morpheme*, the smallest unit determined by the morphological analysis. This helps to recognize the exact boundaries of protein names. Moreover, our morphological analyzer can deal with compounds. This offers a simple way to adapt name descriptions from biomedical resources for language processing. Using GENIA corpus 3.01, our method attains f-score of 70 points for protein molecule names, and 75 points for protein names including molecules, families and domains.

1 Introduction

This paper describes a protein name tagging method which is a fundamental precursor to information extraction of protein-protein interactions (PPIs) from MEDLINE abstracts. Previous work in bio-entity (including protein) recognition can be categorized into three approaches: (a) exact and approximate string matching (Hanisch et al., 2003), (b) hand-crafted rule-based approaches (Fukuda et al., 1998) (Olsson et al., 2002), and (c) machine learning (Collier et al., 2000), (Kazama et al., 2002).

Previous approaches in (b) and (c) ignore the fact that bio-entities have boundary ambiguities. Unlike general English, a space character is not a sufficient token delimiter. Moreover, name descriptions in biomedical resources are mostly compounds. A conventional English preprocessing undergoes a pipeline of simple tokenization and part-of-speech tagging. The tokenization is based on a graphic word[1] for the subsequent part-of-speech tagging to work. The conventional paradigm does not properly handle peculiarities of biomedical English. To remedy the problem, we propose morphological analysis which achieves sophisticated tokenization and adapts biomedical resources effectively.

Our method identifies protein names by chunking based on *morphemes*, the smallest units determined by morphological analysis. We do not use graphic words as a unit of chunking to avoid the **under-segmentation** problem. Suppose that a protein name appears as a substring of a graphic word. Chunking based on graphic words fails, because graphic words are too coarsely segmented. Instead, chunking based on morpheme overcomes the problem, and the exact boundaries of protein names are better recognized.

Below, we describe our method of protein name tagging, including preprocessing, feature extraction (Section 2), and experimental results (Section 3). We mention related work in bio-entity recognition (Section 4) and give concluding remarks (Section 5).

2 Protein Name Tagging

Our task is to identify non-overlapping strings that represent protein names in text. Figure 1 gives an

[1] A graphic word is defined to be a string of contiguous alphanumeric characters with spaces on either sides; may include hyphens and apostrophes, but no other punctuation marks. Quoted from p.125 in Manning and Schütze (1999).

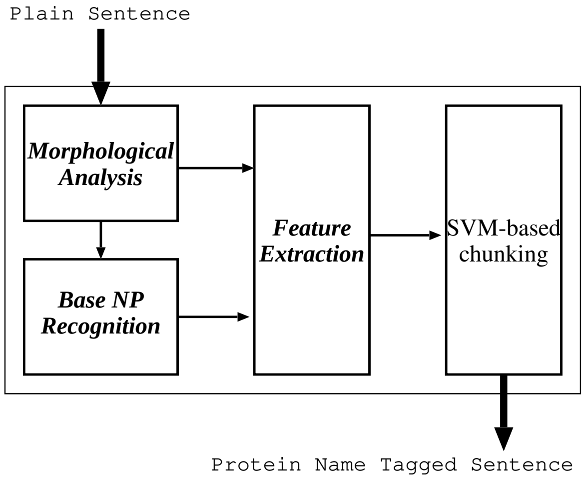

Figure 1: Overview of our protein name tagging method.

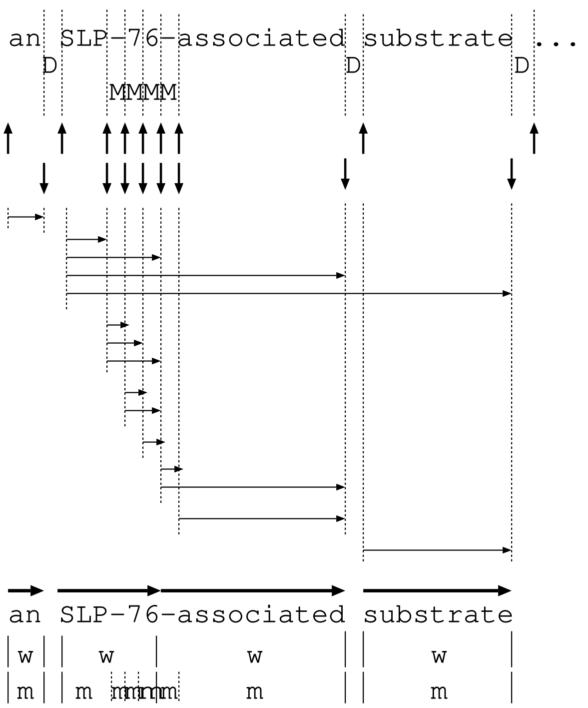

Figure 2: Definition of terms used in this paper. $\uparrow$ is a cps start and $\downarrow$ a cps end. M is a mark and D is a delimiter. The cps starts and cps ends can be determined by marks M and delimiters D. $\rightarrow$ is a token found in the dictionary by common prefix search. A bold $\rightarrow$ is the optimal path in the trellis. w is a word. m is a morpheme.

overview. A plain sentence undergoes morphological analysis and BaseNP recognition. The latter preprocessing is to reflect an intuition that most protein names are found in noun phrases. We extract features from these preprocessing, and represent them as feature vectors. SVM-based chunking is performed using the features to yield a protein name tagged sentence.

2.1 Morphological Analysis

Our morphological analysis gives (a) sophisticated tokenization, (b) part-of-speech tagging and (c) annotation of value-added information such as the stemmed form of a word, accession numbers to biomedical resources. Our morphological analyzer for biomedical English, *cocab*[2], is inspired by the work of Yamashita and Matsumoto (2000).

2.1.1 Preliminaries

We first define terms used in this paper with an illustration in Figure 2.

A *lexeme* is an entry in a dictionary. A *common prefix search* (*cps*) is a standard technique for looking up lexemes in morphological analysis of non-segmented languages. A dictionary is often a trie data structure so that all possible lexemes that match with the prefix starting at a given position in the sentence are retrieved efficiently. A *common prefix search start position* (*cps start*) is a position in a sentence at which a dictionary lookup can start. A *common prefix search end position* (*cps end*) is a position in a sentence by which a matched lexeme must end.

A *token* is a substring in a sentence which matches with a lexeme in the dictionary, and is enclosed by a cps start and a cps end. Note that the matched lexeme is retrieved from the dictionary by common prefix search. A *mark* is a special symbol or substring that by itself can form a token even when it appears within a graphic word. A *delimiter* is a special symbol or code that by itself cannot form a token but can work to delimit tokens. Note also that a delimiter cannot appear on the boundaries of a token, but can appear inside a token. Examples of marks and delimiters are shown in Table 1.

A *word* is a substring in a sentence of which segmentation boundary is determined by the morpho-

[2] http://bmkd.gsc.riken.go.jp/~kaorux/r/cocab.html.en

Table 1: Marks and delimiters used in cocab. Marks include transcription of Greek alphabets that often appear in MEDLINE abstracts.

Delimiter	Mark
space	.,:;"'%/[]{}!?%$&-()
tab	0123456789
CR/LF	alpha beta gamma delta epsilon kappa sigma zeta

logical analysis. A *morpheme* is the smallest unit of a word which is enclosed a cps start and the nearest cps end to the cps start.

The task of morphological analysis is to find the *best* pair $\langle W^*, T^* \rangle$ of word segmentation $W^* = w_1^*, \ldots, w_n^*$ and its parts of speech assignment $T^* = t_1^*, \ldots, t_n^*$, in the sense that the joint probability of the word sequence and the tag sequence $P(W,T)$ is maximized when $W = W^*$ and $T = T^*$. Formally,

$$\langle W^*, T^* \rangle = \arg\max_{\langle W,T \rangle} P(W,T). \tag{1}$$

The approximate solution for this equation is given by

$$\begin{aligned} W^* &= \arg\max_W \max_T P(T|W) \\ &= \arg\max_W \max_T \frac{P(W|T)P(T)}{P(W)} \\ &= \arg\max_W \max_T P(W|T)P(T) \\ &\simeq \arg\max_W \max_T \prod_i p(w_i|t_i)p(t_i|t_{i-2}, t_{i-1}) \end{aligned}$$

and

$$\begin{aligned} T^* &= \arg\max_T P(T|W^*) \\ &\simeq \arg\max_T \prod_i p(w_i^*|t_i)p(t_i|t_{i-2}, t_{i-1}). \end{aligned}$$

2.1.2 Lexeme-based Tokenization

In order to avoid spurious segmentation, we determine cps starts and cps ends in a sentence. Marks and delimiters in a sentence are used to find cps starts and cps ends in the sentence, shown as $\uparrow$ and $\downarrow$ respectively in Figure 2.

Once cps starts and cps ends are determined, the problem is to solve the equation of morphological analysis. It consists of (a) finding a set of tokens that match lexemes in the dictionary, (b) building a trellis from the tokens, and (c) running a Viterbi-like dynamic programming on the trellis to find the path that best explains the input sentence.

In Figure 2, $\rightarrow$ indicates tokens. Both "SLP-76" and "SLP-76-associated•substrate" (• denotes a space character) are tokens since they are lexemes in the dictionary, but "SLP-76-" is not a token since it is not a lexeme in the dictionary. It allows a lexeme-based tokenization which can accommodate a token that is shorter than, the same as, or longer than a graphic word.

The optimal path in the trellis gives a sequence of words that the input sentence is 'best' tokenized and part-of-speech tagged. This is the word-based output, shown as a sequence of w in Figure 2. In addition, our morphological analyzer produces the morpheme-based output, given the word-based output. This is a sequence of the smallest units in each segmented word, shown as a sequence of m in Figure 2. Our chunking is based on morphemes and takes note of words as features to overcome the under-segmentation problem.

2.1.3 Adapting Biomedical Resources

GENIA Corpus 3.0p[3] is used to calculate a word probability $p(w|t)$, and a tag probability $p(t|t', t'')$ which is modeled by a simple trigram. To better cope with biomedical English, we enhance the dictionary (i.e. $p(w|t)$) in a number of ways.

First, we collect human protein names (including synonyms) and their accession numbers from protein sequence repositories, SwissProt (SP) (Boeckmann et al., 2003) and Protein Information Resource (PIR) (Wu et al., 2002). We convert each entry description to a lexeme. A part-of-speech of the lexeme is set to a common noun (NN) where the minimum word probability of NN is assigned for $p(w|t)$. An accession number of the entry is also recorded in the miscellaneous information field of the lexeme. Similarly, Gene Ontology (GO) (Consortium., 2000) terms are converted to lexemes where accession number as well as the root category are kept in the miscellaneous information field. Third, we use UMLS Specialist Lexicon (NLM, 2002) to obtain the stemmed form of a lexeme. A final twist is to associate constituent information for each compound lexeme. A lexeme is compound if

[3]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/

Table 2: A compound lexeme example. “cost” is log of an inverse of $p(w|t)$. “constituents” are obtained from searching single lexemes in the dictionary. “sp” and “pir” are associated with accession numbers. “go” is associated with an accession number and a root category from molecular_function, biological_process, and cellular_component.

key	value	
surface string	ERK activator kinase 1	
cost	10583	
part-of-speech	NN	
reference	sp:Q02750	
stemmed form	–	
constituents	ERK	pir:JQ1400,sp:P29323
	activator	
	kinase	go:016301:molecular_function
	1	

it consists of multiple morphemes, and single otherwise. An example of a compound lexeme is shown in Table 2.

In the conventional paradigm, a token cannot have a white space character. However, 71.6 % of name description in SP are entries of multiple graphic words. This has been a bottleneck in adapting biomedical resources into language processing. In contrast, our morphological analysis can deal with a lexeme with a white space character, and thus offers a simple way to incorporate biomedical resources in language processing. When a sentence is morphologically analyzed, miscellaneous information field is attached, which can be used for the feature extraction component.

2.2 BaseNP Recognition

BaseNP recognition is applied to obtain approximate boundaries of BaseNPs in a sentence. The CoNLL-1999 shared task dataset is used for training with *YamCha*, the general purpose SVM-based chunker[4]. There are four kinds of chunk tags in the CoNLL-1999 dataset, namely IOB1, IOB2, IOE1, and IOE2 (Tjong Kim Sang and Veenstra, 1999). We follow Kudo and Matsumoto (2001) to train four BaseNP recognizers, one for each chunk tag. The word-based output from the morphological analysis is cascaded to each BaseNP recognizer to mark BaseNP boundaries. We collect outputs from the four recognizers, and interpret the tag as outside of a BaseNP if all recognizers estimate the “O(utside)” tag, otherwise inside of a BaseNP. The intention is to distinguish words that are definitely not a constituent of a BaseNP (outside) from words that may be a constituent of a BaseNP (inside). In this way, we obtain approximate boundaries of BaseNPs in a sentence.

[4] http://cl.aist-nara.ac.jp/~taku−ku/software/yamcha/).

Introducing BaseNP recognition as a part of preprocessing is motivated by an intuition that most protein names reside in a noun phrase. Our chunking is based on morphemes. An indication of whether a morpheme lies within or outside a BaseNP boundary seems informative. In addition, the morpheme-based chunking would have narrower local context than the word-based chunking for the same window size. Our intention of approximate BaseNP boundaries is to provide the feature extraction component with the top-down information of morpheme’s global scope.

2.3 Feature Extraction

We extract four kinds of features from preprocessing. Morphological analysis gives information for boundary features, lexical features and biomedical features. BaseNP recognition gives information for syntactic features.

2.3.1 Boundary Feature

Our chunking is based on morphemes of which boundaries may or may not coincide with graphic words. The boundary feature is to reflect the following observation. A general English word tends to have the same morpheme-based segmentation and word-based segmentation, i.e. the degree of boundary ambiguity is low. On the other hand, a protein coding word tends to have different morpheme-based segmentation and word-based segmentation, i.e., the degree of boundary ambiguity is high.

For each morpheme, we have four binary features *lmor*, *ldel*, *rmor*, and *rdel*. *lmor* is 1 if the morpheme is the leftmost morpheme of a word tokenized by the morphological analyzer, and 0 otherwise. *ldel* is 1 if the morpheme is the leftmost morpheme of a graphic word, and 0 otherwise. Similarly, *rmor* is 1 if the morpheme is the rightmost morpheme of a word tokenized by the morphological analyzer, and 0 otherwise. *rdel* is 1 if the morpheme is the rightmost morpheme of a graphic word, and 0 otherwise.

2.3.2 Lexical Feature

The lexical features are multi-valued features. In this work, we consider *part-of-speech*, *stemmed form* and *string* features (e.g. lower-cased string, upper-case letter, numerals, prefix and suffix).

2.3.3 Biomedical Feature

The biomedical feature is designed to encode biomedical domain resource information. The morphological analyzer tokenizes into words with relevant references to biomedical resources. In addition, if the word is derived from a compound lexeme, constituent morpheme information is also attached. (Recall Table 2 for a compound lexeme example.)

The biomedical feature is subdivided into a *sequence* feature and an *ontology* feature. The *sequence* feature refers to a binary feature of accession number reference to SP or PIR. For each word, *sp-word* is set to 1 if the word has an accession number of SP. For each morpheme, *sp-morpheme* is set to 1 if the morpheme has an accession number of SP. *pir-word* and *pir-morpheme* of PIR are the same as those of SP. The *ontology* feature refers to a binary feature of accession number reference to GO. We have *go-word* and *go-morpheme* for GO. Suppose a sentence contains a compound lexeme in Table 2. For the word "ERK activator kinase 1", *sp-word* is set to 1, but *pir-word* and *go-word* are set to 0. For the morpheme "ERK", both *sp-morpheme* and *pir-morpheme* are set to 1, but *go-morpheme* is set to 0.

If *sp-word* or *pir-word* are set to 1, it means that the word exactly matches with a protein name description in SP or PIR. Unfortunately, it is rare due to variant writing of protein names. However, we can expect a sort of approximate matching, by considering morpheme-based features *sp-morpheme* or *pir-morpheme*. Moreover, we add ontology features (*go-word*, *go-morpheme*) in order to obtain thesaurus effects.

2.3.4 Syntactic Feature

The syntactic feature is to reflect an intuition that most protein names are found in noun phrases.

We use two syntactic features, an *indicator morpheme* feature and a *headmorpheme candidate* feature. Both features are relevant only for BaseNP constituent morphemes.

Fukuda et al. (1998) observe that terms such as "receptor" or "enzyme" that describe the function or characteristic of a protein tend to occur in or nearby a protein name. They use those terms as indicators of presence of a protein name. We also express them as a *indicator morpheme* feature, but with an additional constraint that indicators are only influential to morphemes found in the same BaseNP.

In addition, Arabic and Roman numerals and transcription of Greek alphabets are frequently used to specify an individual protein. We call those specifiers in this paper. Without a deep analysis of compound words, it is hard to determine the morpheme that a specifier depends on, since the specifier could be on the left ("alpha-2 catenin") or on the right ("interleukin 2") of the head morpheme. We assume that such specifier morpheme and its head candidate morpheme exist within the same BaseNP boundary and express the observation as the *headmorpheme candidate* feature for each specifier morpheme.

With the absence of a powerful parser, the syntactic features provides only approximation. However, *indicator morpheme* suggests a protein name existence and *headmorpheme candidate* intends to discriminate specifiers appear nearby protein-coding morphemes from the rest.

2.4 Chunking as Sequential Classification

Our protein name tagging is formulated as IOB2/IOE2 chunking (Tjong Kim Sang and Veenstra, 1999). Essentially, our method is the same as Kudo and Matsumoto (2001) in viewing the task as a sequence of classifying each chunk label by SVM. The main difference is that our chunking is based on morphemes, and uses features described in Section 2.3 to serve the needs in protein name tagging.

3 Experimental Results

3.1 Experiment using Yapex Corpus

We first conduct experiments with Yapex corpus[5], the same corpus used in Olsson et al. (2002) to get a direct comparison with the good-performing rule-based approach[6]. There are 99 abstracts for training

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

[6] Olsson et al. (2002) claim they outperform Fukuda et al. (1998) evaluated with Yapex corpus. To date, Fukuda et al. (1998) reports the best result in rule-based approach, evaluated with their closed corpus.

Table 3: Parameter used in the SVM-based chunker YamCha. See (Kudo and Matsumoto, 2001) for more information about parameters.

parameter	description
type of kernel	polynomial
degree of kernel	2
direction of parsing	foreward for IOB2, backward for IOE2
context window	-2 -1, 0, +1, +2
multi-class	one-vs-rest

Table 4: Evaluation criteria used in this paper.

criteria	description
strict	count Correct if the boundaries of system and those of answer matches on Both side.
left	count Correct if the Left boundary of system and that of answer matches.
right	count Correct if the Right boundary of system and that of answer matches.
sloppy	count Correct if any morpheme estimated by system overlaps with any morpheme defined by answer.

and 101 abstracts for testing.

Each sentence undergoes preprocessing, feature extraction and SVM-based chunking to obtain a protein name tagged sentence. We also use YamCha for this task. Parameters for YamCha are summarized in Table 3. Our evaluation criteria follow that of Olsson et al. (2002). We calculate the standard measures of precision, recall and f-score for each boundary condition of *strict*, *left*, *right* and *sloppy* described in Table 4.

The performance of our method on Yapex corpus is summarized in Tables 5 and 6, along with that of Yapex protein tagger.[7]. Our method achieves as good result as a hand-crafted rule-based approach, despite the small set of training data (99 abstracts) which works unfavorable to machine learning approaches. The better performance in *strict* could be attributed to chunking based on morphemes instead of words.

Yapex has a good recall rate while our method enjoys a good precision in all boundary conditions. A possible explanation for the low recall is that the training data was small (99 abstracts) for SVM to generalize the characteristics of protein names. As we will shortly report in the next subsection, we no longer observe a low recall when training with the medium-sized (590 abstracts) and the large-sized (1600 abstracts) data.

[7]Results reported in Olsson et al. (2002) are different from the Yapex web site. Gunnar Eriksson has indicated us to quote the web site as the performance of Yapex protein tagger.

Table 5: Results on Yapex corpus (99 abstracts for training and 101 abstracts for testing). P(precision), R(recall) and F(f-score) are shown. The table shows the protein tagger with an IOB2 chunking with forward parsing.

	Yapex Protein Tagger			SVM (IOB2,forward)		
	P	R	F	P	R	F
strict	0.620	**0.599**	0.610	**0.738**	0.557	**0.635**
left	0.706	**0.682**	0.693	**0.827**	0.625	**0.712**
right	0.749	**0.723**	**0.736**	**0.789**	0.596	0.679
sloppy	0.843	**0.814**	**0.828**	**0.892**	0.674	0.768

Table 6: The table shows the protein tagger with an IOE2 chunking with backward parsing.

	Yapex Protein Tagger			SVM (IOE2,backward)		
	P	R	F	P	R	F
strict	0.620	**0.599**	0.610	**0.738**	0.554	**0.633**
left	0.706	**0.682**	**0.693**	**0.801**	0.602	0.688
right	0.749	**0.723**	**0.736**	**0.797**	0.599	0.684
sloppy	0.843	**0.814**	**0.828**	**0.880**	0.661	0.755

IOB2 chunking with forward parsing gives better results in *left*, while IOE2 chunking with backward parsing gives better results in *right*. The result follows our intuition that IOB2 chunking with a forward parsing intensively learns the left boundary between B(egin) and O(utside), while IOE2 chunking with a backward parsing intensively learns the right boundary between E(nd) and O(utside). Use of a weighted voting of multiple system outputs, as discussed in (Kudo and Matsumoto, 2001), is left for future research.

Effects of each feature in IOB2 chunking with forward parsing are summarized in Table 7. Each feature is assessed by subtracting the focused feature from the maximal model in Table 5. Since the test dataset is only 101 abstracts, it is difficult to observe any statistical significance. Based on the offsets, the result suggests that an incorporation of biomedical features (*sequence* and *ontology*) is crucial in protein name tagging. The contribution of syntactic features is not as significant as we originally expect. Considering syntactic features we use are approximate features obtained from BaseNP boundaries, the outcome may be inevitable. We plan to investigate

Table 7: Effects of each feature contribution on *strict* boundary condition. The F-score is subtracted from the maximal model in IOB2 chuking with forward parsing (Table 5). The upper rows show effects of a single feature removed. The lower rows show effects of multiple features with the same class removed. See Section 2.3 for description of each feature.

feature	F	offset	rank
sequence	0.599	-0.036	1
part-of-speech	0.614	-0.021	2
string	0.615	-0.020	3
ldel and *rdel*	0.628	-0.007	4
indicator term	0.628	-0.007	4
headmorpheme candidate	0.632	-0.003	6
ontology	0.633	-0.002	7
stemmed form	0.634	-0.001	8
biomedical	0.594	-0.041	1
lexical	0.598	-0.037	2
syntactic	0.623	-0.012	3
boundary	0.627	-0.008	4

Table 8: Results on GENIA 1.1 subset of 670 abstracts (590 abstracts for training and 80 abstracts for testing).

	G#protein_molecule			G#protein_X		
	P	R	F	P	R	F
strict	0.657	0.604	0.629	0.694	0.695	0.694
left	0.687	0.632	0.658	0.755	0.755	0.755
right	0.697	0.641	0.667	0.757	0.757	0.757
sloppy	0.727	0.669	0.697	0.827	0.828	0.827

Table 9: Results on GENIA 3.01 set of 2000 abstracts (1600 abstracts for training and 400 abstracts for testing).

	G#protein_molecule			G#protein_X		
	P	R	F	P	R	F
strict	0.711	0.683	0.697	0.757	0.742	0.749
left	0.742	0.712	0.726	0.804	0.788	0.796
right	0.752	0.722	0.737	0.805	0.789	0.797
sloppy	0.787	0.755	0.771	0.858	0.841	0.850

further into effective syntactic features such as word dependency from a word dependency parser.

3.2 Experiment with GENIA Corpus

In order to experiment our method with a larger dataset, we use GENIA corpus 3.01 released recently. Unlike Yapex corpus, GENIA corpus contains 2000 abstracts and uses a hierarchical tagset. For our experiment, we use two definitions for a protein: one to identify `G#protein_molecule` and the other to identify `G#protein_X`. The former is a narrower sense of protein names, and more close to a protein name in Yapex corpus where the protein name is defined as something that denotes a single biological entity composed of one or more amino acid chain. The latter covers a broader sense of protein, including families and domains. We evaluate our method with the two versions of protein names since the desired granularity of a protein name depends on the application.

Two datasets are prepared in this experiment. One is GENIA 1.1 subset and the other is GENIA 3.01 set. The GENIA 1.1 subset contains 670 abstracts from GENIA 3.01 where the same Medline IDs are also found in GENIA corpus 1.1. In addition, we split the GENIA 1.1 subset into the test dataset of 80 abstracts used in Kazama et al. (2002)[8] and the training dataset of the remaining 590 abstracts. The GENIA 3.01 set is an entire set of GENIA corpus 3.01. We randomly split the entire set so that 4/5 of which is used for training the remaining 1/5 is used for testing.

Results in Tables 8 and 9 show that the broader class `G#protein_X` is easier to learn than the narrower class `G#protein_molecule`. Results of protein name recognition in Kazama et al. (2002) using GENIA 1.1 are 0.492, 0.664 and 0.565 for precision, recall, f-score respectively. GENIA 1.1 has only one class for protein name (`GENIA#protein`), while GENIA 3.01 has hierarchically organized tags for a protein name class. Assuming that `GENIA#protein` in GENIA 1.1 corresponds to `G#protein_X` in GENIA 3.01, we could claim that our method gives better results to their SVM approach. The better performance could be attributed to chunking based on morpheme instead of graphic words and better adaptation of biomedical resources. Next, we compare Yapex performance with `G#protein_molecule` trained with 1600 abstracts (cf. Table 5 and Table 9), though tagging policy and corpus are different. Our method significantly outperforms in *strict*, better in *left* and *right*, slightly lost in *sloppy*. With a large dataset of training data (1600 abstracts), we obtain 70 points of f-score for `G#protein_molecule` and 75 points of f-score for `G#protein_X`, which are comparable to approaches reported in the literature.

[8]http://www-tsujii.is.s.u-tokyo.ac.jp/ kazama/papers/testid

An increase of training data from 590 abstracts to 1600 abstracts helps the overall performance improve, given the corpus error is minimized. Our internal experiments with GENIA 3.0 (the version was corrected to GENIA 3.01) reveal that the corpus error is critical in our method. Even corpus errors have been successfully removed, it would not be practical to increase the size of labor-intensive annotated corpus. Use of unlabeled data in conjunction with a small but quality set of labeled data. e.g. Collins and Singer (1999), would have to be explored.

4 Related Work

Tanabe and Wilbur (2002) use a hybrid approach of transformation-based learning (Brill Tagger) with rule-based post processing. An obvious drawback in their approach as with other rule-based approaches including Fukuda et al. (1998) is that the approaches cannot handle many correlated features. As pointed out in their paper, errors in the early stage of rule application are often propagated to the later stage, damaging the overall performance. In contrast, our method can deal with correlated features owing to the generalization characteristic of SVM.

5 Conclusion

This paper describes a method to find protein names by chunking based on a *morpheme*, which leads to better recognition of protein name boundaries. For this, we propose morphological analysis of which core technologies are found in non-segmented languages. With the large dataset (1600 abstracts for training and 400 abstracts for testing in GENIA 3.01), we obtain f-score of 70 points for protein molecule names and 75 points for protein names, including molecules, families, domains etc. The results are comparable to previous approaches in the literature. We focus protein names as a case study. However, given annotated corpus of similar size and quality, the same approach can be applied to other bio-entities such as gene names.

Acknowledgment

We would like to thank Masahi Shimbo of NAIST for his careful review and helpful suggestions. Our appreciation also goes to development teams of Yapex corpus and GENIA corpus to make the shared resources publicly available.

References

B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. 2003. The SWISS-PROT protein knowledgebase and its supplement TrEMBL. *Nucleic Acids Res.*, 31:365–370.

N. Collier, C. Nobata, and J. Tsujii. 2000. Extracting the Names of Genes and Gene Products with a Hidden Markov Model. *COLING*, pages 201–207.

M. Collins and Y. Singer. 1999. Unsupervised Models for Named Entity Classification. *EMNLP-VLC*, pages 100–110.

The Gene Ontology Consortium. 2000. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29.

K. Fukuda, T. Tsunoda, A. Tamura, and T. Takagi. 1998. Toward information extraction: identifying protein names from biological papers. *PSB*, pages 705–716.

D. Hanisch, J. Fluck, HT. Mevissen, and R. Zimmer. 2003. Playing biology's name game: identifying protein names in scientific text. *PSB*, pages 403–414.

J. Kazama, T. Makino, Y. Ohta, and J. Tsujii. 2002. Tuning Support Vector Machines for Biomedical Named Entity Recognition. *ACL Workshop on NLP in Biomedical Domain*, pages 1–8.

T. Kudo and Y. Matsumoto. 2001. Chunking with Support Vector Machines. *NAACL*, pages 192–199.

C.D. Manning and Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

NLM. 2002. UMLS Knowledge Sources. 13th edition.

F. Olsson, G. Eriksson, K. Franzen, L. Asker, and P. Lidén. 2002. Notions of Correctness when Evaluating Protein Name Tagger. *COLING*, pages 765–771.

L. Tanabe and W. J. Wilbur. 2002. Tagging gene and protein names in biomedical text. *Bioinformatics*, 18(8):1124–1132.

E.F. Tjong Kim Sang and J. Veenstra. 1999. Representing Text Chunks. *EACL*, pages 173–179.

C.H. Wu, H. Huang, L. Arminski, J. Castro-Alvear, Y. Chen, Z.-Z. Hu, R.S. Ledley, K.C. Lewis, H.-W. Mewes, B.C. Orcutt, B.E. Suzek, A. Tsugita, C.R. Vinayaka, L.-S.L. Yeh, J. Zhang, and W.C. Barker. 2002. The Protein Information Resource: an integrated public resource of functional annotation of proteins. *Nucleic Acids Res.*, 30:35–37.

T. Yamashita and Y. Matsumoto. 2000. Language Independent Morphological Analysis. *6th Applied Natural Language Processing Conference*, pages 232–238.