

Gene/protein name recognition using Support Vector Machine after dictionary matching

Tomohiro Mitsumori¹, Sevrani Fation¹, Masaki Murata²
Kouichi Doi¹ and Hirohumi Doi¹

¹Graduate School of Information Science,
Nara Institute Science and Technology,
8916-5, Takayama-cho, Ikoma-shi, Nara, 630-0101, Japan
{mitsumor, fation, doy}@is.aist-nara.ac.jp
doi@cl-sciences.co.jp

²Keihanna Human Info-Communication Research Center,
2-2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
murata@crl.go.jp

Abstract

In our experiment, we carried out a gene/protein name recognition using YamCha (Yet Another Multipurpose CHunk Annotator) chunker based on Support Vector Machine algorithm. We considered lexical features (i.e. bag of words, part-of-speech, orthographic, prefix and suffix), and information of gene/protein name dictionary which were collected SWISS-PROT and TrEMBL. Our best results were 0.8245 (precision), 0.7416 (recall) and 0.7809 (balanced f-score).

1 Organization

This paper is organized as follows. In Section 2 we describe our system description. In this section we explain the way of features extraction and machine learning. In Section 3 we describe our result and discussion.

2 System description

Fig. 1 shows our system description. Our system is based on machine learning algorithm. We choose support vector machine (SVM) as machine learning algorithm. Gene and protein names are tagged as positive example in training data. Feature extraction (e.g. bag of words, part of speech, orthographic, prefix and suffix) is carried out. We also use a feature of whether a focused word is included in a part of gene or protein name string (*Yes*) or not (*No*). We call the feature *dictionary feature*. Machine learning is carried out using above training corpus, and support vectors are produced. SVM classification is carried out using the produced support vectors. Finally, gene and protein names are tagged in the test corpus.

2.1 Training data

We used “TAGGED_GENE_CORPUS” as training data. We used BIO representation. This means that beginning word of chunking is tagged *B*, inner word is tagged *I* and others are tagged *O* (see Fig. 3). For example, “*Nerve growth factor*” is tagged such as “*Nerve/B growth/I factor/O*”.

2.2 Feature Extraction

We used the following features such as bag of words, orthographic, POS, prefix, suffix and preceding class. We used the Brill Tagger¹ [4] for POS tagging. The tagger was not trained using biomedical literature, we used an original lexicon.

¹<http://www.cs.jhu.edu/~brill/>

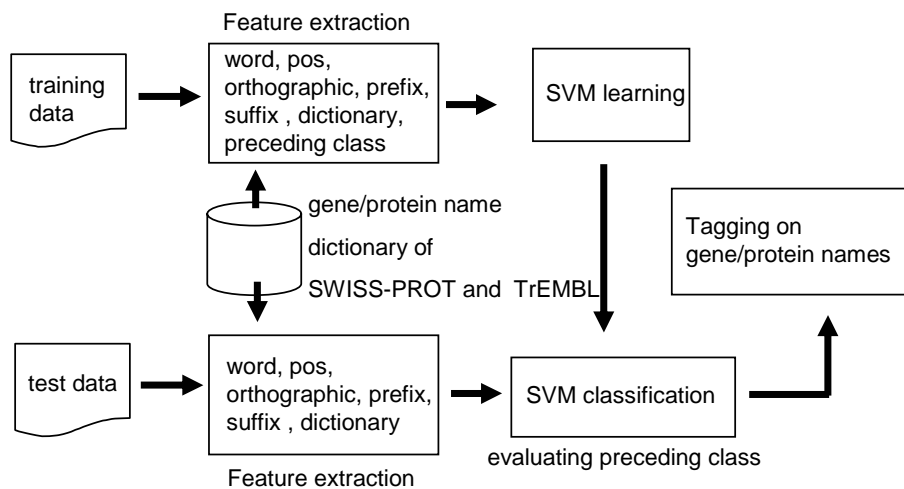


Figure 1: System description. In SVM classify, preceding class feature have to be evaluated dynamically because a preceding class is unknown in test corpus.

Table 1 shows example of the orthographic feature. This orthographic feature are the same as used in Ref. [1]. Prefix feature means uni-gram, bi-gram and tri-gram of beginning letter of word. For example, prefix of “*Nerve*” are “*N*”, “*Ne*” and “*Ner*”. Suffix feature means uni-gram, bi-gram and tri-gram of ending letter of word. For example, suffix of “*Nerve*” are “*e*”, “*ve*” and “*rve*”.

We added a dictionary feature. We made a gene and protein name dictionary which were collected from SWISS-PROT[2] and TrEMBL[2]. There are 96,195 protein names and 115,663 gene names in SWISS-PROT. There are 376,596 protein names and 31,414 gene names in TrEMBL. We made two dictionaries, (1) only SWISS-PROT (We call it Gene/Protein name Dictionary 1 (GPD1)) and (2) SWISS-PROT and TrEMBL (We call it GPD2). If a word is included in the gene and protein name dictionary, it is tagged ‘Y (Yes)’. It is rare to match exactly between words in sentence and protein names[5], because protein names have many variety. So we adopt partial matching to improve matching. Fig. 2 shows an example of uni-gram, bi-gram and tri-gram matching. In this example sentence “*Nerve growth factor (NGF)* “, uni-grams are “*Nerve*”, “*growth*”, “*factor*”, “(”, “*NGF*” and “*)*”. If a uni-gram is used in the GPD, the uni-gram is tagged ‘Y (Yes)’. If a uni-gram is not used the GPD, the uni-gram is tagged ‘N (No)’. We ignored stop words². Bi-grams are as follows, “*Nerve growth*”, “*growth factor*”, “*factor (* Tri-grams are as follows, “*Nerve growth factor*”, “*growth factor (*”, ... We adopt uni-gram, bigram and tri-gram for protein names, and uni-gram for gene names.

We tried to three runs about ways of using gene/protein name dictionary.

- 1st run Exact pattern match between the GPD1 and words in sentence.
- 2nd run Regular expression pattern match between the GPD1 and words in sentence. We regarded as match if non-alphabetical letters are or not included in the strings. For example, “NF-kappa B” is matched to “NF kappa B”, “NFkappa B”, “NFkappaB” and so on.
- 3rd run Exact pattern match between the combination of GPD2 and words in a sentence.

We did not distinguish the capital letter and the lower-case letter above three runs.

Preceding feature means to include BIO classes of preceding words among the feature vector. In our experiment, we include positions from -2 to -1. For example, when we focus the position 0 (see Fig. 3), preceding classes are *B* of position -1 and *O* of position -2. Preceding feature are decided dynamically. Other features are decided statistically.

²<http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhhelp.html>

Table 1: Orthographic feature.

Feature	example	Feature	example
DigitNumber	15	CloseSquare]
SingleCap	M	Colon	:
Greek	alpha	SemiColon	;
CapsAndDigits	I2	Percent	%
TwoCaps	RalGDS	OpenParen	(
LettersAndDigits	p52	CloseParen)
InitCaps	Interleukin	Comma	,
LowCaps	kappaB	FullStop	.
Lowercase	kinases	Determiner	the
Hyphon	-	Conjunction	and
Backslash	/	Other	* + #
OpenSquare	[

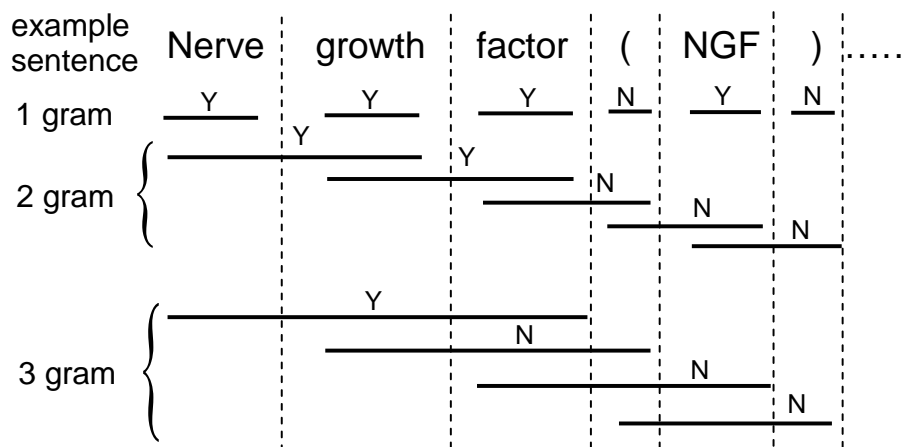


Figure 2: Example of dictionary feature. This is an example sentence “*Nerve growth factor (NGF) ...*”. We adopt uni-gram, bi-gram and tri-gram for protein name and uni-gram for gene name. Y (N) denotes that a gram match (does not match) to a word used in protein or gene name dictionary.

	WORD	POS	ORTHO	PREFIX	SUFFIX	DIC.	TAG
position -3	such	JJ	Lowercase	s su suc	h ch uch	N N N N	O
position -2	as	IN	Lowercase	a as -	s as -	N N N N	O
position -1	NF-kappa	NNP	Greek	N NF NF-	a pa ppa	Y Y N N	B
position 0	B	NNP	SingleCap	B - -	B - -	Y Y N N	I
position +1	that	IN	Lowercase	t th tha	t at hat	N N N N	O
position +2	are	VBP	Lowercase	a ar are	e re are	N N N N	O
position +3	constitutively	RB	Lowercase	c co con	y ly ely	N N N N	O

Figure 3: An example of context window. This is an example of the sentence “... *such as NF-kappa B that are constitutively ...*”.

2.3 Machine Learning

We used the SVM algorithm as machine learning method. Concretely, we use YamCha³ [3] which is SVM-based chunker. The parameters of YamCha is shown in Table 3. Features for SVM learning is shown in Table 2. We used a polynomial kernel of degree two. The direction of parsing was forward (left to right). In Fig. 3, we show an example of a context window which produce feature vectors (large framed box). In this example, I is the estimated tag. In this case, the context window is from position -2 to +2. This means that information of position -2, -1, +1 and +2 is included in feature vectors to decide the class of the focused word (position 0). Tag *B*, *I*, and *O* mean *beginning*, *inner of chunking* and *other term* respectively. For classifying *B*, *I* and *O*, we need to use SVM which expands “multi-class problem”.

Table 2: Features used in the experiment.

Feature	Value
bag of words	all words in training data
orthographic	Capital, Symbol etc.
prefix	From 1, 2 to 3 gram of beginning letter of word
suffix	From 1, 2 to 3 gram of ending letter of word
part of speech	Brill tagger
preceding class	-2, -1
gene/protein name dictionary	protein name collected from SWISS-PROT and TrEMBL

Table 3: The parameters of YamCha.

Parameter	Value
kernel	polynomial
degree of kernel	2
direction of parsing	forward
window position	-2,-1,0,+1,+2
multi-class	pair-wise

3 Results and Discussion

Table 4 shows the results of three runs (case 1) and a case not using dictionary feature (case 2). The best “balanced f-score” was the 2nd run which used the regular expression pattern match with SWISS-PROT. The differences among three runs were less than 1%. In our experiment, 3rd run (using bigger dictionary) was worst. This suggest that a lot of noises were collected in 3rd run.

We compared the results case 1 and case 2 (see Table 4). The best result (2nd) was higher than the result in case 2 about 2.5 %.

We also carried out machine learning using other parameters (e.g. (1) polynomial kernel with degree 3, (2) from position -3 to position +3 as window size and (3) one-vs-rest as multi class method). As a result, the parameters shown in Table 3 was a best set.

³<http://cl.aist-nara.ac.jp/taku-ku/software/yamcha/>

Table 4: Results of our system. Case 1 means cases using dictionary feature. Case 2 means a case not using dictionary feature. TP, FP and FN denote numbers of true-positive, false-positive and false-negative.

	run	Precision	Recall	Balanced f-score	TP	FP	FN
	1 st SWISS-PROT	0.8245	0.7416	0.7809	4412	939	1537
case 1	2 nd SWISS-PROT using regular expression	0.8230	0.7433	0.7811	4422	951	1527
	3 rd SWISS-PROT and TrEMBL	0.8225	0.7408	0.7795	4407	951	1542
case 2	without dictionary feature	0.8122	0.7075	0.7562	4209	973	1740

References

- [1] Takeuchi K. and Collier N. 2003. Bio-Medical Entity Extraction using Support Vector Machine. *In Proceedings of the ACL-03 Workshop on Natural Language Processing in Biomedicine*, 57-64.
- [2] Boeckmann B, Bairoch A, Apweiler R, Blatter MC, Estreicher A, Gasteiger E, Martin MJ, Michoud K, O'Donovan C, Phan I, Pilbout S and Schneider M. 2003. The SWISS-PROT protein knowledge-base and its supplement TrEMBL in 2003. *Nucleic Acids Research*, Vol. 31, No. 1: 365-370.
- [3] Kudo T. and Matsumoto Y. 2001. Chunking with Support Vector Machines. *North American Chapter of the Association for Computational Linguistics(NAAACL)*, 192-199.
- [4] Brill E. 1994. Some Advances in Transformation Based Part of Speech Tagging. *Proceedings of the National Conference on Artificial Intelligence*, AAAIPress: 722-727.
- [5] Yamamoto K., Kudo T., Konagaya A. and Matsumoto Y. 2003. Protein Name Tagging for Biomedical Annotation in Text. *In Proceedings of the ACL-03 Workshop on Natural Language Processing in Biomedicine*, 65-72.