



Using name-internal and contextual features to classify biological terms

Manabu Torii*, Sachin Kamboj, K. Vijay-Shanker

Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA

Received 22 July 2004

Available online 25 September 2004

Abstract

There has been considerable work done recently in recognizing named entities in biomedical text. In this paper, we investigate the named entity classification task, an integral part of the named entity extraction task. We focus on the different sources of information that can be utilized for classification, and note the extent to which they are effective in classification. To classify a name, we consider features that appear within the name as well as nearby phrases. We also develop a new strategy based on the context of occurrence and show that they improve the performance of the classification system. We show how our work relates to previous works on named entity classification in the biological domain as well as to those in generic domains. The experiments were conducted on the GENIA corpus Ver. 3.0 developed at University of Tokyo. We achieve f value of 86 in 10-fold cross validation evaluation on this corpus.

© 2004 Elsevier Inc. All rights reserved.

Keywords: Natural language processing; Bioinformatics; Information extraction; Named entity classification

1. Introduction

Given the wealth of biomedical information that resides in the literature, there has been considerable interest in methods of extracting information from this textual information source. To facilitate this task, there has also been considerable work done recently in recognizing named entities in biomedical text (e.g., [1–6]). Much of this work on named entity recognition has focused on detecting occurrences of protein and gene names in text. This focus on protein and gene name recognition can at least partly be attributed to the importance placed on these entities, the many special technical challenges in recognizing their names, and because much

of the information extraction work has centered around extraction of protein–protein interactions.

There are some important reasons to consider the detection of names of other types of entities as well. As the work on information extraction broadens and is not limited to extraction of protein–protein interactions, it becomes necessary to recognize names of other types of biological entities. Second, classification of names can help improve the precision of the name detection methods. For example, KEX [6] is a protein name recognizer and hence labels each name it detects as a protein. However, names of different types/classes of entities share similar surface characteristics including, for example, the use of digits, special characters, and capitalizations. Due to this reason, KEX and other protein name recognizers can pick names of entities other than proteins (but will label them as proteins). Narayanaswamy et al. [4] reports that precision of protein name detection can be improved by recognizing names

* Corresponding author. Fax: +1 302 831 4091.

E-mail addresses: torii@cis.udel.edu (M. Torii), vijay@cis.udel.edu (K. Vijay-Shanker).

of types other than protein. A third application of classification is in improving coreference resolution, a task of much importance in information extraction. In specialized domains such as biomedicine, the sortal/class information can play an important role for this task. In fact, the coreference resolution method described in [7] seeks to use such information by using the UMLS system [8]¹ and by applying type coercion. Finally, many information extraction methods are based on identifying or inducing patterns by which information (of the kind being extracted) is expressed in natural language text. If we can tag the text with occurrences of various types of names (or phrases that refer to biological entities) then better generalizations of patterns can be induced.

For the task of recognition of names of different types of biological objects, not only should the presence of names be detected, but each of these detected names has to be classified appropriately. That is, we can think that the task of recognizing named entities comprises of two subtasks: named entity detection (i.e., figuring out a string of characters and words makes up a name) and classification (i.e., figuring out the type of object that the detected name refers to). The latter task is the focus of this work. The classification task, whether or not it is performed independently of named entity detection, is an integral part of named entity extraction. The classification task has been an object of study in the natural language processing (NLP) and information extraction communities (not necessarily from the biomedical domain). (see, for example, [9,10] and various approaches reported in the MUC conferences [11]). In our approach, we will examine the classification task, the focus of this work, separate from the name detection task. Like [9,10,12], we believe that the sources of information that help in solving the two tasks are not identical. For instance, part of speech tag information and many orthographic features that are important for the detection task are not useful for the classification task. More importantly, the two tasks and their underlying methods are distinct enough that we believe they can be examined separately. Another reason why we are separating out the classification task from the name detection task is because, for some of the above applications, it is just as important to classify phrases (which are not names) that refer to biological entities and not just the names. In such a scenario, the name detection task itself is no longer important and therefore again the separation of classification from the detection of names is warranted.

In [13], we had previously investigated various sources of information from the point of view of the ex-

tent to which they help with the name classification task. As also in the work of [14], useful sources of information for classification can be *name-internal* as well as *name-external*. In this work, we provide a method for classification largely based on the conclusions we drew from [13]. Both our previous investigation reported in [13] as well as our current work builds upon ideas that are part of a named entity recognizer reported in [4] which is a manually configured rule-based system designed to identify biological names of five different types. For purposes of classification, this system considered the following pieces of information to varying extents: certain words occurring in names, suffix information, certain words appearing in neighboring context, and heuristics based on detection of some syntactic constructions such as conjunction and appositives. In the following sections, we will discuss how we use these sources of information, augment them and consider some additional features, and relate our methods with those of others who might have used similar features.

2. Using an annotated corpus

Most of the features used in [4] were hand-selected and were picked for the five types of entities considered in that work. In this work, we are more interested in the name classification task in general and do not wish to presume our own classification. Additionally, rather than hand select the useful features (as was done in [4]), we would like to select them automatically. We use the GENIA corpus Ver. 3.0 [15] for both these purposes, i.e., to determine the set of types under which the named entities fall as well as for purposes of choosing the name-internal (words, suffixes, and a set of examples) features, and contextual (nearby phrases) features for classification purposes.

In the GENIA corpus Ver. 3.0 (consisting of 2000 MEDLINE abstracts), the identified names are annotated with 36 types. As has been done in previous works [12,16,17], we merge the subtypes of *protein*, *DNA*, and *RNA* in the GENIA ontology to obtain a set of 22 types. This should allow for easier comparison with other results evaluated on the GENIA corpus. There are altogether 75,108 name occurrences in the corpus.

To evaluate our classification methods, we conducted a 10-fold cross validation each time using 1800 abstracts for training and the remaining 200 for testing.

3. Name-internal features

There are essentially two types of name-internal sources of information that we use. The first involves the use of some informative words that indicate the type of a name/phrase and that appear in certain positions

¹ The Unified Medical Language System (UMLS) was developed at the National Library of Medicine, at National Institutes of Health at Bethesda, USA.

within the name. We generalize from this feature and not only do we look at the specific words but also the specific suffixes which provide similar information about a name/phrase's type. While such features do occur fairly often in biological names and are often sufficient for classification when they are present, obviously many terms will not contain such features. For such cases, we consider an example-based approach to classification. If we have a list of example names with their type information, then given a new name, we can try to find examples that closely match the given name and use their classification in our prediction. We now discuss the two approaches that use only the words appearing within the names/phrases, i.e., the approaches based on name-internal features.

3.1. Head-words

This feature borrows from the notion of *f* terms used in [4], which generalizes the *f* terms of KEX [6]. In KEX, certain words such as “gene,” “receptor,” “kinase,” etc., were used to indicate the possible presence of protein/gene names. The named entity recognizer of [4] generalized this idea to select a wider set of these so-called *f* terms, and to not only identify names but to also use their presence for classification purposes. Essentially, their idea is based on two factors. Words such as “kinases,” “proteins,” “receptors,” and “cells” can be thought of as names of classes in an ontology for biological entities. Hence, when these words appear in names, they carry the information necessary for classification purposes. The second aspect that was considered in [4] is that names of biological entities often appear to be like descriptive noun phrases, e.g., “mitogen-activated protein kinase” or “Ras guanine nucleotide exchange factor.” In noun phrases, it is the head-word that usually bears the information about the type of object that the phrase refers to. Further, in English noun phrases, the heads appear to the right. Combining these ideas we can classify “mitogen-activated protein kinase” based on the word “kinase” and classify “Ras guanine nucleotide exchange factor” based on the word “factor” (rather than, say, the word “nucleotide”).

The *f* terms (given the above discussion, we shall simply call them head-words) were manually chosen in [4], and susceptible to errors of omission. Also, our interest is in investigating the classification task in general and not restricting our studies to a specific set of types. Hence, we would like to collect them in an automated fashion. We use the name-annotated GENIA corpus to determine the useful head-words in the following way. Given a word, say *w*, we can collect all the names (in the partition we use for training/development purposes) that have *w* in the rightmost position (to ensure it appears as a head-word), and look at the distribution of the types these names belong to. The more skewed

this distribution is, the more informative the head-word *w* is. In fact, in one training partition, all 665 name occurrences ending with the word “receptor” are of type *protein*. Clearly, the selection of such words is dependent on the corpus and the ontology used. For example, words such as “region” and “domain” that have previously been considered as *f* terms in [4] are less informative given the types used in the GENIA corpus (as compared to the types used in [4]). In one partition, we observed that the distribution for “region” had 268 occurrences of type *DNA* and 72 occurrences of type *protein*.

This implies that the word “region” is not fully informative for the particular types in the GENIA ontology. But GENIA types for “region” are limited to a few. It is conceivable that there can be a plausible type spanning these few GENIA types as subtypes in another ontology. We can get some clue about these subtypes by looking at modifications of the word “region.” Since modifiers usually appear to the left of modified words, we look at the word “region” as well as the word preceding it. We find the occurrences of “promoter region,” “control region,” and “flanking region” are mostly of type *DNA* (*domain or region*). On the other hand, the occurrences of “terminal region” are mostly of type *protein* (*domain or region*). For this reason, we should sometimes consider the final pair of words rather than just the last word.

We said that we identify the head-word by considering the rightmost word in the name. To be precise, for purposes of identification of the “rightmost” words, we strip off a few terms that might appear in the rightmost positions. These include single upper case letters, Roman numerals, Arabic numerals, and Greek letters. The rightmost words are obtained after we strip off such specifiers [2]. Thus, given a name “Janus Kinase 3,” we consider “kinase” as its head-word. We believe this is warranted because the specifiers (by themselves) do not bear useful information for classification purposes.² However, the specifiers can be useful in combination with certain head-words. While such combinations are not very effective with our set of 22 types, they are quite useful if we consider the original 36 types. For example, the appearance of “kinase” as a head-word (when we take all 36 types) was distributed among different protein-related types, but the appearance of “kinase” with a specifier such as a single alphabet letter or a Roman/Arabic numeral is a strong predictor of type *protein molecule*.

We now discuss how we select and use these features for classification purposes. Instead of selecting useful

² Also, as there is not much consistency in hyphenation and spacing, we normalize the names before the specifiers are removed. Then, for example, “Stat3alpha” “Stat3 alpha,” “Stat 3 alpha” are treated alike.

head-words explicitly, we learn a decision list [18] of rules that predict the type. Decision lists have been successfully applied to NLP tasks including word-sense disambiguation (e.g., [19]) and named entity classification (e.g., [9]).

Decision list rules are generated from rule templates. These templates specify the kind of information that may be used for the antecedent of the rules (the conditions). In our method, the rules are conditioned on the head-word or head-word in combination with the previous word and/or the specifier to the right. For instance, the condition part of one of our rule template is “if the head of the name is _____ and the previous word is _____, then ...” where “then ...” will specify the assignment of types to names.

Instead of just assigning one type to a name, our decision list induction program assigns a probability to each type. We choose to do this because we wish to predict the type not just on the basis of one feature but in combination with the decisions based on all the applicable name-internal and external features. The probabilities are estimated using the training partition of the corpus. Some rules generated in this manner are “if the head-word is ‘receptor,’ then $p = 1.00$ for *protein*.” and “if the head word is ‘motif,’ then $p = 0.76$ for *DNA (domain or region)*, $p = 0.20$ for *protein (domain or region)*, $p = 0.02$ for *polynucleotide, ...*” On the other hand, we also have a rule “if the head-word is ‘motif’ and the previous word is ‘octomer,’ then $p = 0.92$, 0.01 , and 0.07 for *DNA*, *protein*, and *polynucleotide*, respectively.” as well as a rule “if the head-word is ‘motif’ and the previous word is ‘binding,’ then $p = 0.70$, 0.24 , and 0.06 for *DNA*, *protein*, and *polynucleotide*, respectively.”

Rules in the decision list are sorted based on the probability assigned to the most likely type according to the estimated probabilities. In the decision list method to classify a name, only the first applicable rule found in the ordered list is used.

Note our rule templates are conditioned on head-words, with/without previous words and with/without specifiers. Specific rules conditioned on more of these items are likely to achieve higher (estimated) precision in type prediction and hence they are usually ranked higher in the ordered list. However, when rules become specific, names in the training partition that meet the rule conditions are fewer, and sometimes become too few to reliably estimate the rule probabilities. Hence, we consider smoothing the estimated probabilities. We smooth the probabilities using linear interpolation, i.e.,

$$\begin{aligned} &P_{\text{smooth}}(\text{type}|\text{prev}, \text{head}, \text{spec}) \\ &= \lambda \times P(\text{type}|\text{prev}, \text{head}, \text{spec}) + (1 - \lambda) \\ &\quad \times P_{\text{smooth}}(\text{type}|\text{prev}, \text{head}), \end{aligned} \quad (1)$$

where we choose λ as

$$\lambda = \frac{\text{Freq}(\text{prev}, \text{head}, \text{specifier})}{1 + \text{Freq}(\text{prev}, \text{head}, \text{specifier})}. \quad (2)$$

That is, the more frequently the conditional events occur, the higher weight we assign to the maximum likelihood element, $P(\text{type}|\text{prev}, \text{head}, \text{spec})$. $P_{\text{smooth}}(\text{type}|\text{prev}, \text{head})$ is calculated similarly from $P(\text{type}|\text{prev}, \text{head})$ and $P_{\text{smooth}}(\text{type}|\text{head})$, and $P_{\text{smooth}}(\text{type}|\text{head})$ is calculated from $P(\text{type}|\text{head})$ and $P(\text{type})$.

Related work. Collins and Singer [9] also considers the presence of specific words within names for classification purposes. But their method just checks whether a specific word is present and does not consider any positional information and whether it appears as the head. Perhaps such information is not relevant in the domain considered in [9], but we believe that it is important in the biomedical domain as stated earlier. Recently Lee et al. [12] also take a similar approach. They consider a list of words and verify whether any of these words appears in one of the 3 rightmost positions (not necessarily the head position). This list of words is taken from the most frequently occurring words in the GENIA corpus. Finally, we would also like to note that [20] also considers head-words for purposes of detecting *binding* terms, i.e., terms that could designate bindable entities. They consider a manually chosen set of words, and any noun phrase having one of these words as its head is then considered as a binding term.

3.2. Suffixes

In the previous section, we considered some words that are highly associated with respective types, and use occurrences of these words at head positions in a name for classification. Hence, they are known to help in predicting entity types. In this section, we discuss the association of a word with a type through its suffix. Consider, for example, the name “erythrocyte.” This can be classified as type *cell* based only on the suffix “cyte.” Similarly, the names “adenovirus” and “papillomavirus” are classified through their suffixes “virus.” For these reasons, we consider suffixes as a generalized notion of head-words, and we detect and use them in a decision list model just like head-words.

Consider a suffix string α . We then consider all names whose head-word (again ignoring specifiers) contain α as a suffix. We can then look at the distribution of the types of these names to determine their utility for classification purposes. As an example, the suffix “ine” is not very informative. In one of the training partitions, 689 names appeared as type *cell line*, 305 as type *protein*, 112 as type *amino acid monomer*, 82 as *other organic compound*, and 19 as *other name*. Now if we extend this suffix to be of length 4, we find many useful suffixes. For example, the suffix “kine” is now closely associated with type *protein*. In fact, 305 of 308 occurrences have this type. The

suffix “line” (as in “cell line”) is almost always associated with the type *cell line*. This shows that there is information gain (the standard measure to decide how to expand node in decision tree induction methods) in going from the suffix “ine” to those that extend it. In general, information gain may be well-suited to determine whether extending a suffix is warranted.

Again, as with head-words, we are not just interested in selecting suffixes. We wish to obtain the decision rules that associate various types with probabilities. Since we wish to treat suffixes in the same way as head-words, the only change we make is that, in the decision list rule templates we replace the attribute “head-word” by “head-word or suffix.” “zyme,” “mycin,” and “amide” are examples of useful suffixes we find.

Related work. Suffixes (and prefixes) are used in [12] for classification purposes. Again, like the choice of words, they appear to consider all of the suffixes from the most frequently occurring words in the corpus. The use of suffixes has also played a part in the analysis of compound words in the biomedical domain as described in [21,22]. Since these works concern analysis of the compound words, not only suffixes but also other morphemes are relevant. Since analysis is carried out to infer the meaning of the compound words, the list of suffixes, prefixes, etc., is manually determined.

3.3. Example-based classification

Biological entities are sometimes named in an ad hoc manner or at least their underlying meaning is not immediately apparent. Names like “anti CD 95 Ab” do not have useful head-words/suffixes to help in the classification task. In such cases, we could attempt to classify the names by matching them with (similar) names in a dictionary, if one were available.

While we cannot expect dictionaries to be constructed for all the types we use in classification, we can collect examples of names for these types. In our case, we use annotated names in the training partition of the GENIA corpus as the examples. For a name (query name) that needs to be classified, we can use similar names from this set of examples to predict the classification of the query name. That is, we are predicting the type of the query name using a nearest neighbor approach.

It is not sufficient to look for an exact name match, since we cannot expect to find the exact same name. However, for classifying the name “CD95,” the presence of other names such as “CD28” can be used to reliably predict the type for the given name. Here, we use approximate matches to find similar names, and use them for classification purposes. Such a method then can be thought of as a way to find generalization from the names in a dictionary/example-based approach.

Usually, to find similar examples, standard string matching algorithms are often used, which produce a

similarity score that varies inversely with the number of edit operations needed to match two strings identically. However, we abandoned the use of standard algorithms as their performance was rather poor for classification purposes. Primarily this was due to the fact that these algorithms do not distinguish between matches at the beginning and at the end of two name strings. In general, words at the left ends of names can be irrelevant to their type identity. On the other hand, the words at the right end bear the semantic type information. For example, the common words in “*transcription factor API*” and “*phorbol ester-inducible transcription factor API*” imply their type identity, but the common words in “*vitamin D receptor*” (*protein*) and “*vitamin D-resistant cell line*” (*cell line*) do not. Hence matches and mismatches at the right end should be given higher weight while computing the similarity scores. For this reason, we have developed our own weighted edit algorithm.

We will first discuss two issues related to tokenization in matching. As mentioned in the detection of the “rightmost” word in head-word/suffix classifier, we detect terms which are simply single alphabets, Greek alphabets, numbers, etc., and call them *specifiers*. In the current context, this is done so that we can differentiate the mismatch of two specifiers from the mismatch of two significant (i.e., non-specifier) tokens. We wish to assign a lower penalty for the former to instantiate our intuition that mismatch of specifiers is not critical for type classification purposes. For example, we believe that the mismatch at the specifier positions in “stat 1 alpha” vs. “stat 3 beta” is not critical for type assignment purposes.

Second, mostly due to the lack of systematic usage, we normalize strings for comparisons purposes. This normalization inserts a blank space in place of hyphens, and before the first numeral and greek alphabets. Thus, our normalization will consider “Stat3beta,” “stat3-beta,” “stat 3beta,” and “stat 3 beta” as identical. Also, similarly in cases like “mAb,” and “IgG,” we insert a blank space before an interior uppercase alphabet to obtain “m Ab” and “Ig G.” This allows for easier comparison between “IgG” and “IgH” as they will now be tokenized as a significant token (“Ig”) followed by a specifier token (“G” or “H”).

We will now describe our version of a simple weighted edit algorithm. An example that illustrates the working of this algorithm is given in Table 1. The two given strings are aligned to match up first the matching significant tokens and then the matching specifier tokens between the pairs of matching significant tokens. Each token is given a position number with the rightmost token getting position 1. The position numbers are incremented by 1 as we go leftwards. Each position, p , is then given a weight, $weight(p)$. The rightmost token is given weight 1 and this is decreased by multiplying with a discount factor (with value less than 1) as we move leftwards. The discounting factor depends on whether the position is occupied by a

Table 1
Matching and score assignment

Position	8	7	6	5	4	3	2	1
Query			<i>anti</i>	<i>CD</i>	<i>95</i>			<i>Ab</i>
Example	<i>cross</i>	<i>linked</i>	<i>anti</i>	<i>CD</i>	<i>3</i>	<i>epsilon</i>	<i>M</i>	<i>Ab</i>
Match score	0	0	1	1	m_1	m_2	m_2	1
Weight	$d_1^3 d_2^4$	$d_1^3 d_2^3$	$d_1^3 d_2^2$	$d_1^3 d_2$	d_1^3	d_1^2	d_1	1

(1) The query “anti-CD95 Ab” and the example “cross-linked anti-CD3epsilon mAb” are tokenized, and then aligned based on the common tokens (position 1, 5, and 6) and on the similar tokens (position 4). (2) A match_score of 1 is assigned to positions 1, 5, and 6, and that of 0 is assigned to positions 7 and 8. Substitution of specifiers occurs at position 4, and hence a match_score of m_1 is used. At positions 2 and 3, m_2 is assigned for the deletion of specifiers. (3) The weight is 1 at the rightmost position 1. The weight at position $p + 1$ is then calculated as d_1 times the weight at p if p is the specifier position, and d_2 times the weight at p otherwise. (4) By setting m_1, m_2, d_1 , and d_2 manually as 0.9, 0.8, 0.9, and 0.5, respectively, the similarity score between this pair of names is 0.87.

specifier or a significant token. Since we take a sequence of specifiers as less important than a sequence of significant tokens, the discount factor associated with specifier, d_1 , is more than that of the discount associated with significant token, d_2 . This way the weight of a token to left of a specifier is more (i.e., discounted less) than if the token to the right is a significant one.

Next, we describe how we assign a match_score for each position depending on what type of edit operation is necessary at that position. An exact match receives a score of 1. The next highest score is given to a mismatch (i.e., requiring a substitution operation) of specifiers (see position 4 in example of Table 1). Call this match_score m_1 . An insertion or deletion of specifier (e.g., position 3 in Table 1) is penalized slightly more. If this match_score is m_2 then $m_1 > m_2$. Finally, substitution of significant tokens or insertion/deletion of significant tokens (e.g., position 7 in Table 1) is penalized heavily. Currently we set the match_score in this case to be zero.

To assign a match score for the query-example candidate pair, we take the weighted sum of match scores at each position, i.e., $\sum_p \text{weight}(p) \times \text{match_score}(p)$. This score is normalized to account for length by dividing it by the score that would have been assigned for a perfect match, i.e., $\sum_p \text{weight}(p) \times 1 = \sum_p \text{weight}(p)$. This gives us our similarity score,

$$\text{Similarity} = \frac{\sum_{p \in \text{position}} \text{weight}(p) \times \text{match_score}(p)}{\sum_{p \in \text{position}} \text{weight}(p)}. \quad (3)$$

Given a query name and some similar examples in the training corpus, likeliness that the name is tagged with a certain type is estimated with the following score:

$$\begin{aligned} \text{Score}(\text{type}|\text{query}) &= \frac{\sum_{e \in \text{example}} \text{Similarity}(\text{query}, e) \times \text{Freq}(\text{type}, e)}{\sum_{t \in \text{all types}} \sum_{e \in \text{example}} \text{Similarity}(\text{query}, e) \times \text{Freq}(t, e)}, \end{aligned} \quad (4)$$

where $\text{Freq}(t, e)$ is the number of abstracts in which the example e is tagged with the type t .

For the efficient retrieval of examples, we use inverse indexation as in document retrieval systems. That is, we

create a hash table associating each significant token with the list of names in the training partition containing that token. Then, given a query name, we use this table to efficiently retrieve names that have common significant tokens. Names that share more tokens with the query will be retrieved first (as they are more likely to have higher similarity scores), and hence we can stop gathering names once we retrieve a sufficient number of candidate examples. Since our matching algorithm emphasizes the rightmost matches/mismatches, candidates with the common words at their left ends will not be selected as similar examples and hence will not have significant role in classification. Hence, to further speed-up the selection of useful candidates we consider an additional heuristic that only uses the hash table for the rightmost two significant tokens.

Finally, because of the example-based method’s time complexity and because of the high precision and recall of the headword/suffix method, we use example-based method only when head-word/suffix rules do not apply.

Related work. The example-based type prediction, using the algorithm above, is a variation of the nearest neighbor method which has been used in named-entity extraction (e.g., [23]). Also, measuring similarities among terms has been used in automatic terminology acquisition in [24]. They determined the similarities based on lexical, as well as contextual and syntactic features. In computing a similarity score based on lexical features, the number of common heads and modifiers between the two terms being compared is considered. Their method assigns higher priority to shared heads over shared modifiers. This might correspond to a bias to the right as in our similarity computation.

4. Context-based classification

Contextual information has been effectively used for the task of word sense disambiguation (WSD) (see, e.g., [19]). For this task, given an occurrence of a word w , the occurrence of other words or phrases nearby can help in disambiguation among the multiple senses

of w . For example, occurrence of the word “manufacturing” nearby can help disambiguate between the factory and the botanical sense of the word “plant.” Such an approach to WSD has been explored in the biomedical domain as well. For instance, in [25], this approach is used to disambiguate ambiguous biomedical terms including abbreviations. Also, in [26], the presence of specific words and phrases nearby is used to disambiguate between the “gene,” “mRNA,” and “protein” senses of a name (since the same name can be used for a gene or its products). For example, [26] detects that the word “encode” occurring nearby implies a higher likelihood of the name in question being used in the “gene” sense rather than the “protein” or the “mRNA” sense.

The work of [26] can also be thought of as type classification of a name (where there are three types gene/mRNA/protein). In fact, the use of nearby words for classification in the named entity extraction has been carried out before (e.g., [9,10]).

In employing this approach to WSD or type classification, in the named entity extraction task, the notion of “nearby” has to be fixed. In [19], it is observed that words that are close by tend to have stronger predictive power for WSD. Hence, one approach is to consider a method using few strong context evidences given by closely co-occurring phrases (as in [19]). An alternate approach used for WSD task has been to use phrases that can appear further away and then combine these multiple, possibly weak ones as in [27]. Previously, we implemented the former approach with the decision list learning algorithm and the latter approach with Naïve Bayes method to combine the multiple evidences [13] and found the latter to be more effective. But in this work, we found the former approach to be more effective. This can perhaps be attributed to the fact that in this work (in contrast to the work reported in [13]) with an increased number of types and hence an increased number of name occurrences packed close to each other, clues that are farther away tend to be ineffective. Hence, we report our results only for the approach of considering words and phrases that are in close proximity to a name occurrence. We induce a decision list (using the annotated named entity corpus for training purposes) to learn which phrases are important and how they can be used to assign types to name occurrences.

The decision list rules are obtained by instantiating templates that predict the type of a name occurrence by conditioning it on a phrase and its position relative to the occurrence of this named entity. The templates consider the co-occurring phrase to be either a single word or two consecutive words. The relative position part of the rule condition specifies directionality (i.e., whether the phrase appears to the left or right of the name) and distance (in terms of number of words from the name). From the directionality information, the set of rules can be separated into two lists: for the left and

right contexts. As before, the precision is used to order the rules obtained by instantiating the rule templates.

As before, the induced rules also assign probabilities to each type so that they can be combined with different sources of information. The conditional probability of a type given the co-occurring phrase, direction, and distance is estimated from the training partition

$$P(\text{type}|\text{context}) = \frac{\text{Freq}(\text{type}, \text{context})}{\text{Freq}(\text{context})}, \quad (5)$$

where “context” consists of the three conditioning variables. Smoothing is done the same way as described in Section 3.1. We apply a frequency threshold of 5 (i.e., $\text{Freq}(\text{context}) \geq 5$) for the rules (i.e., if the condition for a rule is found to occur 4 or less times in the training corpus, it is not included in the decision list).

Common words such as “and,” “can,” “is,” “therefore,” “of,” etc., are unlikely to aid in classifying names effectively. We noticed that sometimes (in experiments where no or low precision thresholds were set) the application of rules involving such common words leads to errors (due to the preponderance of protein type in the corpus leading to rules that predict this class). To avoid this situation, we manually choose a small set of *stop words*. The decision rule learner has been implemented to ignore these words for use as contextual phrases while instantiating the rule templates.

Some of the strong evidences for classification found in the left context in this manner are “replication of” for type *virus* and “phosphorylation of” for *protein*. Those found in the right context are “secretion” for *protein* and “replication” for *virus*. The phrase “production in” in the left context is not a strong evidence for any one type, but it narrows down the candidate types to *cell type* and *cell line*.

Related work. In the Biomedical domain, context information has already been used for name classification (e.g., [4,12,20,26]). However, contextual evidences are limited to few manually selected phrases with directional (left/right) information in [4,20]. Our decision list approach allows us to automatically learn significant phrases directly from the training data. Our approach is in this sense closer to [26] who use context to disambiguate between protein/gene/mRNA senses of a name. While we have used a completely supervised approach using the GENIA corpus for training, [26] do not use any such annotated corpus but instead use a bootstrapping approach like [19]. Previously, in [28], we have experimented with bootstrapping for alleviating the amount of type annotated data, and we will discuss this aspect again in the conclusion section.

5. Some additional sources

We have so far discussed the main features we use for classification purposes. In addition to these we have

considered a few simple heuristics to improve the performance. These are discussed in this section.

5.1. Using acronyms

In biological texts, the use of acronyms and abbreviations is common. Classifying such terms is more difficult than classifying their corresponding long forms for two reasons. The first reason is that abbreviations often lack name-internal clues, namely head-words and suffixes, which their corresponding long forms may have. In such case, we have to rely only on example-based method or on the contextual clues (that are less reliable than name-internal clues in general). The second reason for the difficulty is that different names may share the same alias, and hence the occurrence of such an alias may introduce ambiguity in the sense, e.g., “EMCS” may stand for “extraskelatal myxoid chondrosarcomas,” “emergency medical centers,” or “*N*-(6-maleimidocaproyloxy) succinimide,” each of which is of a different type. However, authors often provide long forms corresponding to their short forms through parenthetical expressions, e.g., “*extraskelatal myxoid chondrosarcomas (EMCs) are characterized by...*” When available, we will exploit this information, and use long forms for the classification of the short form occurrences within one abstract. Identification of corresponding long and short forms has been studied in this domain (among others, in [29–31]). In our simple identification algorithm, we look for a word sequence preceding the parenthesis, which spans the character sequence in the parenthesis, in a manner similar to that described in [32] to identify the long and short form pair. Since our name-internal classification methods place significance toward the right end of names, we are not particular about identification of exact long forms (exact left ends; the right ends are assumed to be the words preceding the parenthetical expressions).

If a short form-long form pair is identified in a document (abstract), we use the classifications based on the long form for the name-internal classification of the other occurrences of the short form. We found that this heuristic helps considerably. In one particular test set partition, there were 1284 instances of acronyms whose long-forms were also available. The head-word/suffix method was able to classify 1083 of these instances correctly. In contrast, without using the head-word/suffix method, but using context, only 322 were classified, of which 167 classifications were correct.

5.2. One sense per abstract

In [19], “one sense per document” heuristic was shown to improve the WSD performance. While a word might have several senses, this heuristic was proposed based on the observation that the occurrences of a word in a single document tends to take on the same sense.

Similarly, we employed “one type per abstract” heuristic after we confirmed its validity over some part of the GENIA corpus. This is assumed implicitly in our acronym strategy discussed before, i.e., while an acronym might have multiple long forms, our strategy assumes that once a long form-short form pair is found in an abstract, all other occurrences of the short form have the same long form in the given abstract.

To apply the “one type per abstract” heuristic, we tried two variations: the overwriting and the non-overwriting methods. First, our type classification method is applied so that each occurrence of a name is assigned a score for each type. We then compute the average of the scores over all the occurrences of the name for each type, and overwrite their original scores with the average. Since all the occurrences of the name now share the same scores for the types, this is analogous to the heuristic of [19] in the sense that the heuristic can modify the previously assigned types (overwriting). Alternatively, in the non-overwriting mode, the averaged score from all the occurrences that have been classified is used only for those occurrences that have not been classified. Note that some occurrences may not be assigned any score if no rule exceeding precision/frequency threshold (discussed in Section 6) applies. In a technical sense, this mode does not correspond to the one-type-per-abstract heuristic as the name occurrences which are assigned types initially are not over-written and hence different occurrences of the same name can be assigned different types. This mode of application is used to increase the ability to assign types to those occurrences which do not seem to have sufficiently strong evidences to be assigned types.

5.3. Coordination/appositive heuristics

Besides the one-type-per-abstract heuristic, we also considered assigning the same type to the names that appear in a coordination construct. This heuristic hypothesizes that names that are grammatically coordinated³ or two phrases that are in an appositive construct⁴ tend

³ When two or more associated phrases are connected with commas and a conjunction (e.g., “and,” “or,” “and/or,” “as well as”) in a sentence, they can be regarded as the coordinated names. For example, in the sentence “... several transcriptional factors including *Sp1*, *AP-1*, *AP-2*, and *NF-kB*,” four names “*Sp1*,” “*AP-1*,” “*AP-2*,” and “*NF-kB*” are regarded as coordinated names. However, in the sentence, “In human monocytes, interleukin 1beta protein production and steady state mRNA levels are ...” while the name “human monocytes” is followed by a comma, the two names “human monocytes” and “human monocytes” will not be considered as coordinated names.

⁴ Many times, two noun phrases in a relation of apposition appear next to each other (separated by a comma), the inserted phrase can be regarded as an appositive. For example, in the sentence “*A human T lymphoid cell line, PEER, dies by apoptosis in...*,” two names “*human T lymphoid cell line*” and “*PEER*” are recognized to be in an appositive construct. Like our coordination extraction program, our appositive detection program falsely extracts in some cases.

to share the same type. These heuristics, however, have a difficulty in two respects. First of all, simple automated methods to identify an appositive or coordination construct can make a few mistakes. Second, the coordination hypothesis especially does not hold true in some cases. For example, consider “*Furthermore, both transforming growth factor and genistein, a tyrosine kinase inhibitor, would suppress IL-2 and IL-12 signaling. . .*” The two names “transforming growth factor”, and “genistein” receive different types in the GENIA corpus. In some sense, one can consider the coordination hypothesis does work here as well if we are willing to consider them both of the type: *objects that affect protein signaling!* But in general, one can see that the result of the application of this hypothesis depends on the particular level of abstraction in the types being considered as well as how the types are carved up in the underlying ontology.

This heuristic may still be useful and improve the recall of classification without hurting the precision much if we apply it only to those names that are classified by the other methods with little confidence. This is because the hypothesis will probably be better than a random guess. Hence, we adopt this heuristic without the overwriting mode, i.e., average scores do not overwrite the classification scores assigned by high confidence rules.

6. Results

Before we discuss our results, we introduce the *classification confidence value*. Let us assume that, for a particular occurrence of a name, scores are assigned to types by one of our classification methods. Then the classification confidence value is simply the ratio of the highest score for one type to the second highest score for another type, i.e.,

Confidence(name instance)

$$= \frac{\text{Score}(\text{type}_1|\text{name instance})}{\text{Score}(\text{type}_2|\text{name instance})}, \quad (6)$$

where $\text{Score}(\text{type}_1|\text{name instance}) > \text{Score}(\text{type}_2|\text{name instance}) > \text{Score}(\text{type}_3|\text{name instance}) > \dots$. Here, $\text{Score}(\text{type}_1|\text{name instance})$, $\text{Score}(\text{type}_2|\text{name instance})$, $\text{Score}(\text{type}_3|\text{name instance})$, \dots are the scores for type_1 , type_2 , type_3 , \dots given the name instance.

If this value is greater than a threshold value, the name can be classified as the type associated with the highest score by the classification method, else the name will not be classified. Higher threshold for the classification confidence implies that the classifiers find the top choice significantly more likely than the next choice. Thus, the instances tagged with higher confidence levels are more likely to be correct. But with a lower threshold for confidence value, more instances are likely to be tagged. To see the trade-off between precision and recall, we present the results in this section for different threshold values.

The classification methods we have discussed exploit complementary features. Hence, these methods can probably be combined to achieve better performance than the individual ones. We combine two methods at a time and derive a combined system at the end. We consider three ways to combine the individual classifiers: *Average*, *Weighted Sum*, and *Max*. In *Average* method, the average of scores assigned by two classifiers is computed for each type and used for classification. In the *Weighted Sum* method, we calculate the weighted sum of scores and then normalize (so that the scores for each type sums to 1.0). In *Max* method, for each instance, the scores assigned by the classifier that has a higher classification confidence are adopted.

6.1. Name-internal methods

We first present the results for the name-internal classification. These results, as with the rest of the results in this section, are given for classification confidence value thresholds of 1, 3, 5, 7, and 10. We can notice that the head-words/suffix classifications tend to be highly accurate with the precision dropping only slightly with lower confidence value thresholds. However, the gain in recall is more significant with the lowering of this threshold (see Fig. 1).

Since we apply the example-based method only to those names that are not classified with head-words/suffixes⁵, a name occurrence is classified by one or the other method only. Since the example-based method also has high precision, we see that the example-based method boosts the recall with no significant change to the precision. While the example-based method achieves high precision on this corpus, the question of its generalizability beyond the use of the GENIA corpus needs to be examined.

6.2. Context-based method

Next, we consider the combination of the classifications by the left context and the right context decision lists. We can see from Fig. 2 that the right context decision lists performs worse than the left-context decision list. For both of these decision lists, the recall is low and the trade-off between precision and recall is sharp for different confidence value thresholds.

For a query name, if only one of these decision lists provides an applicable rule, then obviously the scores associated with the rule were adopted for the name occurrence. We tested the *Average* and *Max* when both of them have rules that applied, and find *Max* marginally outperforms *Average*. In the remaining discussion,

⁵ Note that even if a classification is suggested for an instance, it is not considered as classified unless the confidence value for the instance exceeds the confidence value threshold.

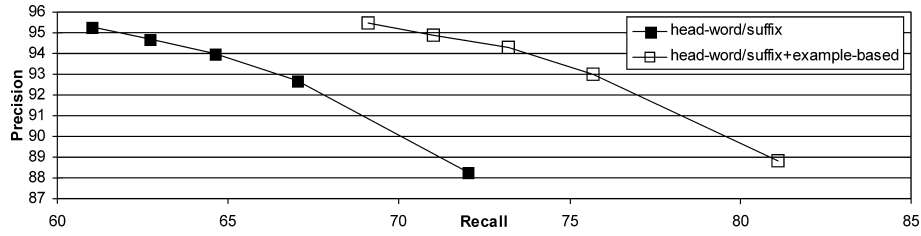


Fig. 1. Evaluation of name-internal classifiers.

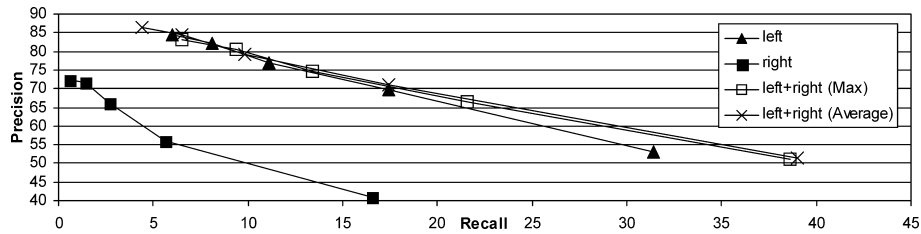


Fig. 2. Evaluation of name-external classifiers.

therefore, we take Max as the combination method for the name-external based classifications.

6.3. Name-internal and external clues

The name-internal and the name-external classifications derived as above are next combined using the Weighted Sum. The weights for the classifier decisions were chosen manually. Given the results for the two types of classifications, we did not experiment by trying to assign a higher weight to the name-external method. The weight ratios tested are 1 to 1 (Average), 3 to 1, and 10 to 1 for the name-internal and the external based classifications, respectively. As it is expected, higher

weights for the name-internal classification improves the results of the combined system more (Fig. 3). However, increasing the ratio beyond 10:1 did not make a difference. Hence, we employ the ratio 10 to 1 in the remaining experiments.

6.4. Classification heuristics

For the classifications obtained above, we applied the one-type-per-abstract heuristic (OTA), exploring both the overwriting and non-overwriting modes. As can be seen from Fig. 4, the improvements are more noticeable at lower confidence thresholds. Our hypothesis is that at higher thresholds levels, most of the clas-

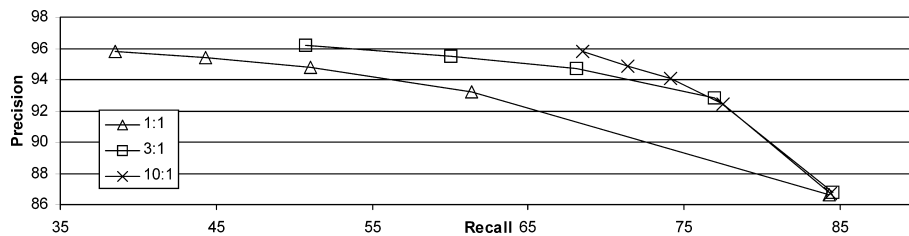


Fig. 3. Combination of name-internal and name-external classifiers.

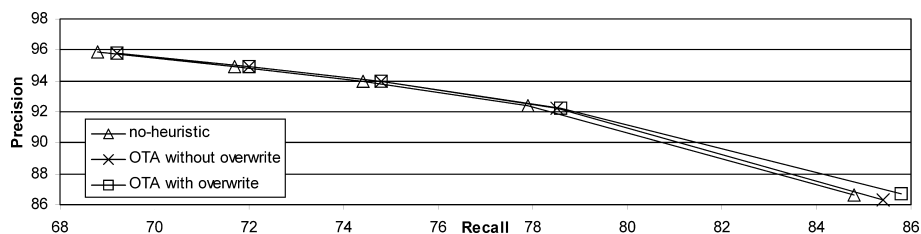


Fig. 4. Application of one type per abstract (OTA) heuristic.

sification is due to the more precise name-internal methods and that because of our use of acronym expansion, the classification should be made independent of position in the abstract. That is, if a name instance is classified confidently by a name-internal method (especially given the high weight assigned to name-internal classifiers), all occurrences of the same name in that abstract should also be classified (not giving scope for new instances being classified by this heuristic) and that the classification will be the same (thereby not giving scope for overwriting as well). In contrast, at lower threshold levels, the name-external classifier comes into play more and now the position of occurrence becomes more relevant. This gives scope for more occurrences to be tagged (as in some positions there may be sufficient contextual clues for tagging and in some other places there may not be) as well as improvements due to overwriting. If the classification is done with lower confidence, it might be safer to go with the classification in majority of the cases. We notice that the heuristic with the overwriting scheme improves the recall by one percent for confidence threshold of one.

Finally, we examined the heuristic based on coordination and appositive constructs. As mentioned previously, we did not explore the overwrite option. This means that the heuristic is applied to a relatively small number of cases. Hence we present the evaluation in the form of a table (Table 2) focusing only on the cases where it was applied.

Table 3 shows the precision and recall of the integrated system for individual types.

7. Discussion

The results in the previous section suggest that the features we have considered for the task of type classification are indeed effective for this task. The name-internal features not only allow for high precision classification, but also are sufficient for the system to achieve high recall. This bears out our hypothesis that the head-words (a generalization of the *f* term idea used in [4,6]) are useful in this domain for classification purposes and that suffixes could also play a similar role. We have also noted that when an acronym is used and its long form is defined in an abstract, the long form often carries useful head-word/suffix to enable us to classify acronyms effectively.

Of course, a number of names will not have useful head-words/suffixes and for these cases we attempt an example-based classification, a nearest-neighbor method, using the training part of the corpus to provide the examples. Clearly, as with any nearest neighbor approach, the quality of predictions depends on the set of examples available. So the generalizability of this method can be questioned. The abstracts in the GENIA corpus represent a collection which is perhaps not wide ranging in topics. So, when this method is used on names found in abstracts on other topics, using names from the GENIA corpus may not be effective. Clearly, additional work will be required for collecting a set of examples in such cases. If the set of topics are widened, this method may become more susceptible to names that might represent entities of different types. However, we would like to point out that for acronyms (which are often cited as examples of such polysemy) for which the

Table 2
Application of one type per coordination/appositive heuristic

Confidence	1	3	5	7	10
No. Applied	33	120	136	147	156
No. Correct	23	77	87	92	97

Table 3
Evaluation of the combined classifier

Confidence	Freq.	1			3			5			10		
		pre.	rec.	<i>f</i> val.	pre.	rec.	<i>f</i> val.	pre.	rec.	<i>f</i> val.	pre.	rec.	<i>f</i> val.
Total	7511	87	86	86	92	80	85	93	76	84	95	71	81
Protein	2501	89	93	91	93	87	90	94	83	88	96	76	85
DNA	858	88	82	85	91	76	83	92	71	81	95	61	74
Cell type	620	76	93	84	89	63	74	92	56	70	93	48	63
Other organic compound	391	83	60	70	89	55	68	92	49	64	94	45	61
Cell line	334	80	55	65	88	49	63	89	45	60	92	42	57
Multi cell	179	88	83	85	92	79	85	93	77	84	95	72	82
Lipid	151	87	86	87	92	83	87	93	82	87	95	78	85
Virus	103	95	81	88	97	80	87	97	77	86	98	77	86
Cell component	85	91	77	84	92	75	82	94	68	79	95	65	78
RNA	62	85	70	77	91	67	77	94	65	77	94	54	68

long form-short form is given in the abstract, as discussed above, this issue is not likely to arise.

As is to be expected, we also found the example-based/nearest-neighbor method to be most time-consuming among all classifiers that use the different information sources. For one of the partitions with 200 test abstracts, the application of this classifier took 25 s⁶ compared to 35 s it takes for all other methods described above put together. This is despite the fact that we spent more time on trying to make its implementation efficient and did not make similar efforts for the others.

For the contextual features, we found that the left-context phrases are often more available and more reliable than the right-context. We find Max as the best way of combining them. This should yield the same results as having just one decision list combining left and right contextual rules as the decision list orders the rules by their precision. While the recall of the head-words/suffixes was more than we originally anticipated, the precision and recall of the contextual features were not as much as we expected based on our previous experience. We believe that this is due to the large number of types being considered in this corpus. As a result, many of the names found in each abstract are in close proximity to each other. The latter might cause some of the otherwise useful clues to become less effective. The former might imply that the amount of training data we have from the GENIA corpus may not be sufficient. In a previous work [28], we had obtained better results using contextual features in this domain. In that experiment, we were trying to classify among five types only and also the amount of training data used was an order larger. To obtain the training data, we applied a bootstrapping approach (similar to [19]) starting from some examples that were tagged with high confidence. Similarly, we would like to explore the bootstrapping method by applying our high confidence rules learned from the GENIA corpus on a large amount of un-annotated text, and iterating the process of learning.

We have experimented with other features and applied different heuristics with varying degrees of success. We treat certain types of strings (sequence of digits, single upper case letters, Greek alphabets, etc.) differently during identification of useful head-words and suffixes as they do not help in classification, in the templates of name-internal rules, and during example-based classification (penalizing their mismatches less) in order to obtain generalization beyond these examples. As noted, detection of the long form of an acronym is useful. The one-type-per-abstract in overwriting mode and coordination rule in non-overwriting mode also help us achieve better results especially in terms of recall.

Since we focus exclusively on the classification task, it is hard to compare our results with those obtained by others. There have been several efforts to apply named-entity extraction work to the GENIA corpus. [12,16,17] have in fact used the same types as in this work. Also, the results reported by [12] are also based on 10-fold cross-validation like the results reported in this paper. There are a number of other similarities with our work as well. Among them is the separation of classification from name detection/identification. Many of the features used are also common although their use of words found in the name is not limited to rightmost words. Also, their choice of word and suffix features is based on frequency of occurrence and a machine learning method (SVM in their case) uses them in making the classification decision. As far as we can tell, they have not employed acronym detection and used different heuristics such as one-type-per-abstract, coordination, etc. While they have provided the name identification and classification results separately, the results are not directly comparable. At a first glance, the numbers we got appear to be better, but one must also keep in mind that the names they classify are the ones their name identifier detects, and it is not clear how much error in the detection phase propagates to the classification phase.

In this work, we have manually assigned weights while combining the classifiers using various different features. Clearly, it would have been preferable to automatically decide how to combine these disparate sources. In fact, we explored some automated ways of assigning the weights following methods discussed in [32]. (For example, one way to determine weights is to use the precision of individual classifiers, where the precision is determined based on a development set and weight the more precise ones more than the less precise ones). But we found that the results from these methods did not fare as well as our best manual choice of weights.

More recently, we tried to use maximum entropy method [33] using the exact same features that we use here, using the examples-based method only when the decision-list approach would use it too (i.e., when there were not any head-words/suffix for classification). Maximum entropy models will assign weights to each feature according to their effectiveness. Yet the principled method of assigning weights underlying maximum entropy modeling did not yield better classification results. We are currently looking into slightly varying the selection of features that might suit maximum entropy modeling better.

8. Conclusion

In this work, we investigate the named entity classification task in the biology domain. We believe that various language processing methods and applications for

⁶ The experiment was conducted on a Sun Blade 1000 workstation with a 750 MHz UltraSPARC-III and 1GB memory.

this domain can be enhanced by the ability to classify a detected term into different types. We apply machine learning methods to build a classifier and examine features based on words appearing within the names as well as features based on neighboring context. For the former, we show how to automatically select useful words as well as suffixes and apply decision-list learning method to induce rules for classification. We formulate a new string-similarity scoring method to build a nearest-neighbor based classification module. For features appearing outside the name, we first apply decision list learning method to automatically learn useful phrases appearing to the left or right of a name occurrence for classification purposes. Finally, we evaluate the effectiveness of some novel strategies we propose based on the context of use including the detection of acronym-full form pair, and detection of some syntactic constructs (coordination and appositives). We show how these features can be combined to produce state-of-art classification results. The training and testing are conducted on the GENIA corpus Ver 3.0 developed at the University of Tokyo.

Acknowledgments

We thank the anonymous reviewers for their comments and are grateful for their numerous suggestions to improve the paper. We thank K.E. Ravikumar and Meenakshi Narayanaswamy of AU-KBC Research Center, Chennai, for sharing their programs and for many discussions. Many of their ideas form the starting basis of this work. We thank the members of the GENIA project at the University of Tokyo for making the corpus available and for patiently answering all our questions.

References

- [1] Collier N, Nobata C, Tsujii J. Extracting the names of genes and gene products with a Hidden Markov Model. In: Proceedings of the international conference on computational linguistics; 2000. p. 201–7.
- [2] Franzén K, Eriksson G, Olsson F, Asker L, Lidén P, Cöster J. Protein names and how to find them. *Int J Med Inf* 2002;67(1–):49–61.
- [3] Hanisch D, Fluck J, Mevissen HT, Zimmer R. Playing biology's name game: identifying protein names in scientific text. *Pac Symp Biocomput* 2003;403–14.
- [4] Narayanaswamy M, Ravikumar KE, Vijay-Shanker K. A biological named entity recognizer. *Pac Symp Biocomput* 2003;427–38.
- [5] Proux D, Rechenmann F, Julliard L, Pillet V, Jacq B. Detecting gene symbols and names in biological texts: a first step toward pertinent information extraction. In: *Genome inform ser workshop genome inform*, vol. 9; 1998. p.72–80.
- [6] Fukuda K, Tsunoda T, Tamura A, Takagi T. Information extraction: identifying protein names from biological papers. *Pac Symp Biocomput* 1998;707–18.
- [7] Castaño J, Zhang J, Pustejovsky J. Anaphora resolution in biomedical literature. In: *Int symp on reference resolution*; 2002.
- [8] Unified Medical Language System, National Library of Medicine. Available from: <http://www.nlm.nih.gov/research/umls/umlsmain.html>.
- [9] Collins M, Singer Y. Unsupervised models for named entity classification. In: *Joint SIGDAT conference on EMNLP and VLC*; 1999.
- [10] Cucerzan S, Yarowsky D. Language independent named entity recognition combining morphological and contextual evidence. In: *Joint SIGDAT conference on EMNLP and VLC*; 1999.
- [11] The Sixth Message Understanding Conference (MUC6). San Mateo, CA: Morgan Kaufman; 1995.
- [12] Lee KJ, Hwang YS, Rim HC. Two-phase biomedical NE recognition based on SVMs. In: *Proceedings of the ACL-03 workshop on NLP in biomedicine*; 2003. p. 33–40.
- [13] Torii M, Kamboj S, Vijay-Shanker K. An investigation of various information sources for classifying biological names. In: *ACL-03 workshop on NLP in biomedicine*; 2003. p. 113–20.
- [14] McDonald D. Internal and external evidence in the identification and semantic categorization of proper names. In: Boguraev B, Pustejovsky J, editors. *Corpus processing for lexical acquisition*. Cambridge, MA: MIT press; 1996. p. 21–37.
- [15] Kim JD, Ohta T, Tateisi Y, Tsujii J. GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics* 2003;19(Suppl. 1):180–2.
- [16] Kazama J, Makino T, Ohta Y, Tsujii J. Tuning support vector machines for biomedical named entity recognition. In: *ACL-02 workshop on NLP in biomedical domain*; 2002. p. 1–8.
- [17] Zhou G, Zhang J, Su J, Shen D, Tan C. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics* 2004;20(7):1178–90.
- [18] Rivest RL. Learning decision lists. *Mach Learn* 1987;2(3):229–246.
- [19] Yarowsky D. Unsupervised word sense disambiguation rivaling supervised methods. In: *ACL-95*; 1995. p. 189–96.
- [20] Rindfleisch TC, Hunter L, Aronson AR. Mining molecular binding terms from biomedical text. In: *Proceedings of the AMIA symposium*; 1999. p. 127–31.
- [21] Baud RH, Rassinoux AM, Ruch P, Lovis C, Scherrer JR. The power and limits of a rule-based morpho-semantic parser. In: *Proceedings of the AMIA symposium*; 1999. p. 22–6.
- [22] McCray AT, Browne AC, Moore DL. The semantic structure of neo-classical compounds. In: *Proc of Annual Symp on Comp Appl in Med Care*; 1998. p. 165–8.
- [23] Sang ETK. Memory-based named entity recognition. In: *CoNLL-2002*; 2002. p. 203–6.
- [24] Nenadic G, Spasic I, Ananiadou S. Automatic discovery of term similarities using pattern mining. In: *Proceedings of CompuTerm 2002*; 2002. p. 43–9.
- [25] Liu H, Johnson SB, Friedman C. Automatic resolution of ambiguous terms based on machine learning and conceptual relations in the UMLS. *J Am Med Inform Assoc* 2002;9(6):621–36.
- [26] Hatzivassiloglou V, Duboue PA, Rzhetsky A. Disambiguating proteins, genes, and RNA in text: a machine learning approach. *Bioinformatics* 2001;17(Suppl. 1):S97–S106.
- [27] Gale W, Church KW, Yarowsky D. A method for disambiguating word senses in a large corpus. *Comput Humanities* 1992;26:415–39.
- [28] Torii M, Vijay-Shanker K. Using unlabeled MEDLINE abstracts for biological named entity classification. In: *Proceedings of genome informatics workshop*; 2002. p. 567–8.
- [29] Liu H, Aronson AR, Friedman C. A study of abbreviations in MEDLINE abstracts. In: *Proceedings of AMIA Symp*; 2002. p. 464–8.

- [30] Liu H, Friedman C. Mining terminological knowledge in large biomedical corpora. *Pac Symp Biocomput* 2003:415–26.
- [31] Schwartz AS, Hearst MA. A simple algorithm for identifying abbreviation definitions in biomedical text. *Pac Symp Biocomput* 2003:451–62.
- [32] van Halteren H, Zavrel J, Daelemans W. Improving accuracy in word class tagging through the combination of machine learning systems. *Comput Linguist* 2001;27(2):199–230.
- [33] Baldrige J, Morton T, Bierner G. The OpenNLP Maximum Entropy Package. Available from: <http://maxent.sourceforge.net/>