A Hybrid Approach to Protein Name Identification in Biomedical Texts

Kazuhiro Seki and Javed Mostafa

Laboratory for Applied Informatics Research, Indiana University

1320 East Tenth Street, LI 011, Bloomington, Indiana 47405, USA

Tel: 812-855-2849

Fax: 812-855-6166

Email: {kseki,jm}@indiana.edu

Acknowledgment

We would like to thank Dr. Damir Ćavar for his helpful comments and the members of the *Proteinhalt i text* project for sharing their invaluable resources for public use. We also wish to thank the anonymous reviewers for their constructive comments. This project is partially supported by the NSF ITR grant #9817572.

Abstract

This paper presents a hybrid approach to identifying protein names in biomedical texts, which is regarded as a crucial step for text mining. Our approach employs a set of simple heuristics for initial detection of protein names and uses a probabilistic model for locating complete protein names. In addition, a protein name dictionary is complementarily consulted. In contrast to previously proposed methods, our proposed method avoids the use of natural language processing tools such as part-of-speech taggers and syntactic parsers and solely relies on surface clues, so as to reduce the processing overhead. Moreover, we propose a framework to automatically create a large-scale corpus annotated with protein names, which can be then used for training our probabilistic model. We implemented a protein name identification system, named PROTEX, based on our proposed method and evaluated it by comparing with a system developed by other researchers on a common test set. The experiments showed that the automatically constructed corpus is equally useful in training as compared with manually annotated corpora and that effective performance can be achieved in identifying compound protein names with PROTEX.

Keywords

Named-entity recognition, protein names, information extraction

1 Introduction

Reflecting the ever-growing digitized publications, enormous efforts have been made for automatically discovering novel information from texts, i.e., text mining. The popularity of text mining techniques is reflected in the large number of publications on this topic found in established forums such as ACM SIG on Knowledge Discovery in Data and Data Mining (KDD) (Zaki et al., 2003) and IEEE International Conference on Data Mining (ICDM) (Cercone et al., 2001).

Text mining is crucial also in the field of cellular and molecular biology because of a strong

demand for discovering molecular pathways, protein-protein interactions, and so on in the literature, which are even for human experts, labor-intensive and time-consuming. Therefore, much research has been conducted to explore text mining techniques on biomedical texts (Adamic et al., 2002; Friedman et al., 2001; Fu et al., 2003; Hirschman et al., 2002; Ng & Wong, 1999; Palakal et al., 2002; Proux et al., 1998; Sekimizu et al., 1998; Thomas et al., 2000).

Our ultimate goal is to realize an automated system to mine novel information from the biomedical literature; specifically, we aim to develop a system to identify relations and interactions between proteins and cancer, that are expected to facilitate the development of new drugs and treatments peculiar to cancer. To accomplish our goal, we start with identifying protein names appearing in biomedical texts. This task can be considered as named-entity recognition partly explored in Message Understanding Conferences (MUCs) (Grishman & Sundheim, 1996), in which proper nouns, such as names of people and companies, locations, and time expressions appearing in newspaper articles were targeted for automatic recognition. The most successful system achieved the F-score of 93.4% (Marsh & Perzanowski, 1998) and the task, in this particular domain, could be seen as a solved problem.

However, identifying protein names is still an open question. This is partially because there are no common standards or fixed nomenclatures for protein names that are followed in practice (Bruijn & Martin, 2002). As new proteins continue to be discovered and named, predefined protein name dictionaries are not necessarily helpful in identifying new protein names. Additionally, protein names frequently appear in shortened, abbreviated, or slightly altered forms (e.g., the use of capital and small letters and hyphens is often inconsistent). Therefore, even the protein names that are already known and are to be included in a dictionary might be overlooked due to the way they are actually written. Another challenging issue in identifying protein names is to find their name boundaries. According to our preliminary research on 99 MEDLINE abstracts (obtained at *Proteinhalt i text* project Web page (Franzén, 2003)), 42% of protein names are composed of multiple tokens (i.e., compound names), and these tokens include symbols, common

nouns, adjectives, adverbs, and even conjunctions, which makes it difficult to distinguish protein names from the surrounding texts (Tanabe & Wilbur, 2002).

Motivated by the above background, we explore a method to identify protein names in biomedical texts with an emphasis on finding compound names. As a final model, we propose a hybrid method taking advantage of hand-crafted rules, probabilistic models, and a protein name dictionary as a complementary means. In addition, we propose a framework to train probabilistic models on an automatically constructed large-scale corpus, where manually annotated corpora are no longer needed for training.

The rest of this paper is structured as follows: Section 2 summarizes past research related to protein name identification. Section 3 explains our proposed approach in detail. In Section 4, the methodology of evaluation is described and the result is presented and discussed. Lastly, Section 5 and Section 6 conclude this paper with our findings and future directions.

2 Related Work

There have been several attempts to develop techniques to identify protein names in biomedical texts. They roughly fall into three approaches, that is, dictionary-based, heuristic rule-based, and statistical.

An approach based exclusively on a dictionary is not necessarily helpful for identifying protein names because new protein names continue to be created and there are often many variations in the way identical proteins are referred to. To tackle this problem, Krauthammer et al. (2001) proposed an approach to protein and gene name identification, using BLAST (Altschul et al., 1997), a DNA and protein sequence comparison tool. Their basic idea involves performing approximate string matching after converting both dictionary entries and input texts into nucleotide sequence-like strings, which can be then compared by BLAST. For evaluation, they extracted gene and protein names from GenBank (Benson et al., 2003) to create a name dictionary, converted

them to nucleotide sequence-like strings according to a predefined conversion table, and applied their method to a review article, which is manually annotated with 1,162 gene and protein names by biology experts. They reported that 409 names out of 1,162 names (35.2%) were not contained in the dictionary and, among them, 181 (44.3%) were fully or partially identified by their proposed method.

Fukuda et al. (1998), Franzén et al. (2002), and Narayanaswamy et al. (2003) proposed rule-based approaches. They exploited surface clues for detecting protein name fragments (i.e., parts of protein names) and used a part-of-speech tagger and/or a syntactic parser for finding protein name boundaries. Typically, the surface clues include the following features, where bold characters indicate the corresponding examples.

- Capital letters (e.g., **ADA**, **CMS**)
- Arabic numerals (e.g., ATF-**2**, CIN**85**)
- Roman alphabets (e.g., Fc **alpha** receptor, 17**beta**-estradiol dehydrogenase)
- Roman numerals (e.g., dipeptidylpeptidase **IV**, factor **XIII**)
- Words appearing frequently in protein names (e.g., myelin basic **protein**, PI 3-**kinase**, nerve growth **factor**)

Franzén et al. (2002) conducted experiments that compared their system (Yapex) with Fukuda's system (Kex) on 101 MEDLINE abstracts. Yapex achieved a recall of 61.0% and a precision of 62.0% as compared to a recall of 37.5% and a precision of 34.3% on Kex in terms of exact match. Incidentally, Narayanaswamy et al. (2003) reported to have achieved a recall of 69.1% and a precision of 96.9% on 55 MEDLINE abstracts, where the precision is much higher than both Yapex and Kex. However, they cannot be directly compared since Narayanaswamy et al. targeted both protein and gene names (where detected names are regarded as correct if they are either gene or protein names), while Yapex and Kex focused specifically on protein names, and their test data are different.

Statistical approach has made a considerable impact on natural language processing (NLP)

research and related areas, such as part-of-speech (POS) tagging, parsing, and speech recognition. In the bioinformatics domain, for example, Collier et al. (2000), Nobata et al. (1999), and Kazama et al. (2002) applied statistical methods (e.g., hidden Markov models, decision trees, and support vector machines) for detecting and classifying gene and gene product names including proteins. The features used in their methods are mostly the same as those used in rule-based approaches, that is, surface clues and parts of speech.

Comparing rule-based and statistical approaches, rule-based approaches have an advantage in a sense that rules can be flexibly defined and extended as needed, but manually analyzing targeted domain texts and crafting rules are often time-consuming. Statistical approaches are relatively easy to apply if appropriate models and training data are given. However, creating training data (i.e., corpora annotated with protein names in this case) requires domain experts and is also time-consuming, and an insufficient amount of training data leads to the data sparseness problem. In general, to achieve higher performance, more complex, elaborated models are desirable, which usually require more training data in order to reasonably estimate the increasing number of parameters.

Rule-based and statistical approaches are, of course, not necessarily exclusive. For instance, Tanabe & Wilbur (2002) proposed a method for identifying gene (and protein names) using both heuristic and statistic strategies. They introduced a new (pseudo) POS tag indicating gene names and trained a rule-based POS tagger on a corpus manually annotated with gene names using the new tag. They used the Brill tagger (Brill, 1994), which does error-driven learning to induce rules for tagging parts of speech. After training on gene names, the tagger can be used for identifying potential gene names as if they are one of parts of speech. The initial result is then filtered to remove misdetected candidates and to identify overlooked names using manually created rules. Lastly, a naïve Bayes classifier is applied to predict the likelihood that each input document does contain gene names. Their experiments demonstrated that higher performance can be achieved for gene/protein name identification on the documents with higher Bayesian scores.

Although there have been much research on protein name identification as shown above, there are no common data set or standard evaluation criteria by which different methods can be fairly compared. To advance the techniques of protein name identification, it is important to have common basis for comparing results as attempted in MUCs (Grishman & Sundheim, 1996). As for data set, there are a few annotated corpora publicly available. One is the GENIA corpus (Ohta et al., 2002), which consists of 2,000 MEDLINE abstracts (version 3.01) and has been annotated with not only proteins but a subset of the substances and the biological locations involved in reactions of proteins. Another corpus was made by the *Proteinhalt i text* (protein concentration in text) project (Franzén, 2003), which is composed of 200 MEDLINE abstracts and was manually annotated with protein names. As for evaluation criteria, Olsson et al. (2002) proposed four criteria to assess the performance of protein name identification, which are explained in Section 4 in more detail.

It should be also noted that, in order to provide a common data set and evaluation criteria to compare different approaches, there is an ongoing project, called BioCreAtIvE (Critical Assessment of Information Extraction systems in Biology) (Hirschman & Blaschke, 2003). The tasks of the project include gene/protein name identification, where each research group independently develops their method to automatically identify gene/protein names, and their results are to be evaluated on a common test set and evaluation criteria. A workshop reporting the results will be held in Spring 2004.[1]

In this paper, we propose a hybrid approach utilizing heuristic rules for initial detection of protein name fragments and a probabilistic model focusing on determining name boundaries. In addition, a protein name dictionary is complementarily consulted. The rules used are simple and easy to develop and the probabilistic model incorporates word classes based on suffixes and word structure, so as to overcome the data sparseness problem. For evaluation, we use the data set produced by the *Proteinhalt i text* project and their proposed evaluation criteria, which enables us to directly compare our method with the previous work conducted by Franzén et al. (2002).

3 Our Proposed Method

3.1 Overview

Figure 1 depicts the framework of our proposed method to be described in this section.

Insert Figure 1 about here

Before entering in protein name identification processes, an input text is first partitioned into sentences and then tokenized, where tokens are defined as words and symbols. For instance, `PI 3-kinase` will be separated into four tokens, i.e., `PI`, `3`, `-`, and `kinase`. Then, we identify protein names through three steps shown in Figure 1. Firstly, fragments of protein names are detected by a set of heuristic rules relying on surface clues which are commonly used for protein name identification. Secondly, because the detected fragments does not usually form complete protein names, their name boundaries are expanded/determined based on either a set of rules or a probabilistic model so as to locate complete protein name candidates. In the case where a probabilistic model is used, a filter is then applied to the protein name candidates to rule out unlikely candidates.[2] Lastly, a protein name dictionary is consulted to detect the protein names that are not recognized by the previous steps. Each step is explained in Section 3.2–3.4 in greater details.

3.2 Protein Name Fragment Detection

To detect protein name fragments, we use several heuristic rules which were derived from previous studies (Franzén et al., 2002; Fukuda et al., 1998; Narayanaswamy et al., 2003) and our preliminary observation on 99 MEDLINE abstracts (Seki & Mostafa, 2003). These abstracts were manually annotated with 1,745 protein names by *Proteinhalt i text* project (Franzén, 2003) and were used as a reference corpus for developing the Yapex protein name tagger (see Section 2).

Words satisfying any of the following conditions are detected as potential protein name fragments.

- Words that include capital letter(s)
- Words that include combinations of Arabic numerals and lower case letters (e.g., p35, cdk5)
- Words with suffixes that often appear in protein name fragments (*–nogen*, *–ase*, and *–in* are considered in this study)
- Words that often appear as protein name fragments (*factor(s)* and *receptor(s)* are considered in this study)
- Roman alphabets that often appear as protein name fragments (*alpha*, *beta*, *gamma*, *delta*, *epsilon*, and *kappa* are considered in this study)

These conditions unfortunately also detect words that are not protein name fragments. For example, if we extract all words containing capital letters, words located in the beginning of sentences will be inevitably extracted as protein name fragments. To decrease these errors, we exclude the following.

- Words with a capital letter in the beginning followed by more than three lower case letters (e.g., According, Basically)
- Words composed of only capital letters longer than six characters (e.g., KTPGKKKKGK)
- Only one character (e.g., A, B, C, $\cdots$)
- Measuring units: *nM*, *mM*, *pH*, and *MHz* (predefined based on the reference corpus)
- Chemical formulas: *CaCl2*, *NH2*, *Ca2*, *HCl*, and *Mg2* (predefined based on the reference corpus)
- Words included in a stopword list. In this study, we used the PubMed Stopword List,[3] which contains 133 function words

Henceforth we call the set of rules described above the "*detection rules*."

3.3 Protein Name Boundary Expansion

Since the *detection rules* detect only protein name fragments in principle, the name boundaries of the detected protein name fragments need to be expanded so as to identify complete protein names. For this purpose, we explore the use of heuristic rules and a probabilistic model. The heuristics were derived from our preliminary observation on the reference corpus (99 MEDLINE abstracts) as used in Section 3.2.

Previous work (Franzén et al., 2002; Fukuda et al., 1998; Nobata et al., 1999) typically made use of POS taggers and/or syntactic parsers for the purpose of determining protein name boundaries, assuming that POS tags and/or noun phrases are good indicators for protein name boundaries. However, according to our preliminary investigation on the reference corpus, protein name fragments can be symbols, nouns, adjectives, adverbs, verbs, and conjunctions; thus, POS tags may not be very informative for the purpose of detecting protein names and their name boundaries. Therefore, our method, unlike previous work, avoids the use of those NLP tools, which results in reducing both processing overhead and the potential number of probabilistic parameters to estimate.

3.3.1 Heuristics

The words and symbols matching the conditions described below are regarded as parts of protein names, and the protein name boundaries are expanded so as to include the matching words or symbols.

The following are the conditions of rules that expand name boundaries rightward, where italic characters denote the protein name fragments detected by the *detection rules*, and the bold characters denote the expanded parts.

- A hyphen (optional) plus a numeral or capital letters less than three characters (e.g., *ATF* **-2**)
- A capital letter in parentheses (e.g., *Ruk* **(I)**)

The conditions of the rules that expand name boundaries leftward are shown below. Words shown in parentheses as instances of frequent words/suffixes are the complete sets considered in these rules.

- A numeral or capital letters less than three characters and optionally a hyphen (e.g., **N -** *glycanase*)
- In the case where the protein name fragment detected by the *detection rules* has a suffix "–in" (e.g. protein):
  - frequent words (binding, related, associated) preceded by a hyphen plus any words (e.g., parathyroid **hormone - related** *protein*)
- In the case where the protein name fragment detected by the *detection rules* has a suffix "–ase" (e.g., kinase):
  - frequent suffixes (–ine, –tide, –yl) optionally followed by a hyphen and numerals (e.g., **tyrosine** *kinase*)
  - frequent suffixes (–one, –sitol) optionally followed by capital letters and a hyphen (e.g., **glutathione S -** *transferase*)

Finally, the following name fragments that were detected but not expanded by the above rules are discarded, as they are unlikely to be protein names without any qualifiers.

- protein, –ase, receptor(s), factor(s), alpha, beta, gamma, delta, epsilon, kappa, I, II, III

Henceforth we call the set of rules described above the "*expansion rules*."

3.3.2 Probabilistic Models

The creation of the *expansion rules* is a human process and it is based on close study of the reference corpus. Hence, certain degree of arbitrariness cannot be avoided. We introduce an alternative approach based on a probabilistic model to learn word collocation patterns without human intervention. Below, we will explain the details of our model with an example context of

"`...with dipeptidyl peptidase IV promoter constructs...`", where "`dipeptidyl peptidase IV`" is the actual protein name and "`peptidase`" and "`IV`" are to be detected by the *detection rules*.

First, we will look at the process to expand name boundaries leftward. Let w_i denote one of the protein name fragments detected in the previous initial detection step. Given a fragment w_i, the probability that a token w_{i-1} immediately preceding w_i is also a protein name fragment can be expressed as a conditional probability $P_p(w_{i-1}|w_i)$, assuming a first-order Markov process. Likewise, the probability that w_{i-1} is *not* a protein name fragment can be expressed as $P_n(w_{i-1}|w_i)$. For the above example, `dipeptidyl` and `peptidase` correspond to w_{i-1} and w_i, respectively. (Notice that `IV` will not be w_i since `peptidase`, the left side of `IV`, is already known (detected) as a protein name fragment.) Thus, we estimate the probabilities P_p(`dipeptidyl`|`peptidase`) and P_n(`dipeptidyl`|`peptidase`).

Based on these probability estimates, we decide whether to expand protein name boundaries. In the case where there is no name boundary between w_{i-1} and w_i (i.e., w_{i-1} is also a protein name fragment), $P_p(w_{i-1}|w_i)$ is expected to be greater than $P_n(w_{i-1}|w_i)$. Thus, we regard w_{i-1} as a protein name fragment if the following condition holds:

$$P_p(w_{i-1}|w_i) > P_n(w_{i-1}|w_i) \quad (1)$$

However, estimating these probabilistic parameters will require a large amount of training texts annotated with protein names, which are expensive to create. To make matters worse, simply using a large-scale corpus cannot be a substantial solution due to the characteristics of protein names: new protein names continue to be created. Previously unseen data could be fatal for probability estimation such as maximum likelihood estimation.

To remedy the data sparseness problem, we generalize words (tokens) to word classes. Table 1 shows some examples. They are automatically and uniquely assigned to each word according to the algorithm presented as a pseudo code in Figure 2.

Insert Table 1 about here

Insert Figure 2 about here

The algorithm determines a unique word class for each word as follows. Given an input word *w*, it first checks whether *w* is included in the predefined set of Roman alphabets, Roman numerals, or punctuations; whether it is a number; or whether it contains one or more capital letters. If so, its corresponding word class is assigned. If not, it then looks at the characters composing *w*. In the case where *w* is composed of alphabets, either *"word"* or one of *"suffix"* classes is assigned, which is described in more detail next. Otherwise, class *"symbol"* is given.

For word *w*, its suffix class (e.g., *suffix_in* and *suffix_ase*) is dynamically generated by extracting a sequence of a vowel, one or more consonants (if any), and either a vowel or a consonant from the ending of *w*. However, class *"word"* will be assigned in cases where the suffix was not extracted (i.e., s_w = `NULL`) or the resulting suffix is longer than the remainder of the word in length. For example, given an input "`peptidase`", it takes "`a`" as a vowel, "`s`" as one or more consonants, and "`e`" as a vowel or a consonant, forming a suffix "`ase`". For another example, suppose "`case`" as input. In this case, again "`ase`" will be extracted as its suffix. However, because the length of "`ase`" (=3) is longer than that of the remainder "`c`" (=1), the suffix will be rejected and class *"word"* will be assigned.

Integrating the word classes to the probabilistic models, we formalize the bigram class models as in Equation (2), where c_i denotes the word class of w_i. The probabilities initially

presented are now defined as a product of class-class and class-word transitional probabilities.

$$P_p(w_{i-1}|w_i) = P_p(w_{i-1}|c_{i-1}) \cdot P_p(c_{i-1}|c_i)$$
$$P_n(w_{i-1}|w_i) = P_n(w_{i-1}|c_{i-1}) \cdot P_n(c_{i-1}|c_i) \quad (2)$$

For our example, these are:

$$P_p(\texttt{dipeptidyl}|\texttt{peptidase}) = P_p(\texttt{dipeptidyl}|\textit{suffix\_yl}) \cdot P_p(\textit{suffix\_yl}|\textit{suffix\_ase})$$
$$P_n(\texttt{dipeptidyl}|\texttt{peptidase}) = P_n(\texttt{dipeptidyl}|\textit{suffix\_yl}) \cdot P_n(\textit{suffix\_yl}|\textit{suffix\_ase}) \quad (3)$$

The probabilistic parameters can be estimated based on corpora annotated with protein names. However, the models still contain raw words w_{i-1}, which may cause the data sparseness problem. To cope with it, we make use of a smoothing method in estimating the transitional probabilities. Preliminarily, we compared Good-Turing estimation (Good, 1953) and Witten-Bell smoothing (Witten & Bell, 1991). Briefly, the former estimates the probabilities based on adjusted word frequencies assuming that their distribution is binomial, and the latter estimates the probabilities by considering how often a class in question has new words (which is assumed to be the number of words associated with the class). For this study, we utilize the Witten-Bell smoothing method, which found to perform better on the reference corpus. Equation (4) shows the definition of Witten-Bell smoothing (for this particular model) to estimate a transitional probability from class c to word w:

$$P_{WB}(w|c) = \begin{cases} \dfrac{F(w,c)}{F(c)+T(c)} & \text{if } F(w,c) > 0 \\ \dfrac{T(c)}{F(c)+T(c)} & \text{otherwise} \end{cases} \quad (4)$$

where $F(x)$ denotes a frequency of x, and $T(c)$ denotes the number of word types which belong to class c.

Likewise, we adopt the model described above to expand/determine protein name boundaries rightward as well. The probability functions are formalized as in Equation (5), where w_{i+1} and c_{i+1} denote the token immediately following the detected protein name fragment w_i and

its word class, respectively.

$$P_p(w_{i+1}|w_i) = P_p(w_{i+1}|c_{i+1}) \cdot P_p(c_{i+1}|c_i)$$
$$P_n(w_{i+1}|w_i) = P_n(w_{i+1}|c_{i+1}) \cdot P_n(c_{i+1}|c_i) \quad (5)$$

For each of the protein name fragments detected by the heuristic rules described in Section 3.2, we compute the probabilities above to determine whether to expand their name boundaries. When expanded, the probabilistic models are iteratively applied to the expanded fragments to further expand name boundaries as long as the condition in Equation (1) holds. For our example,

$$\begin{aligned} P_p(\texttt{dipeptidyl}|\texttt{peptidase}) &= P_p(\texttt{dipeptidyl}|\textit{suffix\_yl}) \cdot P_p(\textit{suffix\_yl}|\textit{suffix\_ase}) \\ &= 0.02912594 \times 0.05825189 \\ &= 0.00169664 \\ P_n(\texttt{dipeptidyl}|\texttt{peptidase}) &= P_n(\texttt{dipeptidyl}|\textit{suffix\_yl}) \cdot P_n(\textit{suffix\_yl}|\textit{suffix\_ase}) \\ &= 0.00040621 \times 0 \\ &= 0 \end{aligned} \quad (6)$$

Because $P_p > P_n$ for this word collocation, "`dipeptidyl`" is also regarded as a protein name fragment. Then, the window will move one word leftward and the same procedure is applied so as to determine whether to include the preceding word (i.e., `with`) as a part of the protein name.

3.4 Filtering

Our ultimate goal is to automatically discover novel information associated with proteins and cancer from the literature, where protein name identification is a fundamental factor whose performance will strongly influence the following processes. Although high recall and high precision are ideal, generally there is a trade-off between them. In this context, it is desirable if we could arbitrarily choose which metric is (and how much it is) preferred in the output of protein name identification (e.g., high recall with low precision, high precision with low recall, or

balanced) depending on reliability of the information we are seeking. This can be done by restricting the system output based on some certainty measure that indicates the extent to which the detected protein names are likely to be actual protein names.

We are currently using certainty score $C(\cdot)$ defined as in Equation (7), where $w_1 \cdots w_n$ denotes a sequence of tokens detected as a protein name, and $F(w_i)$ and $F_p(w_i)$ denote a frequency of w_i in training data and a frequency of w_i which appears as a protein name fragment, respectively.

$$C(w_1 \cdots w_n) = \frac{1}{n}\sum_{i=1}^{n} P_c(w_i)$$

$$\text{where } P_c(w_i) = \begin{cases} \dfrac{F_p(w_i)}{F(w_i)} & \text{if } F(w_i) \geq 3 \\ \dfrac{F_p(c_i)}{F(c_i)} & \text{otherwise} \end{cases} \tag{7}$$

$P_c(w_i)$ is a probability that word w_i is a protein name fragment, and $C(w_1 \cdots w_n)$ is an average of all probabilities associated with w_1 to w_n. Probability $P_c(w_i)$ will take a greater value in the case where a token w_i is predominantly used as a protein name fragment in training data since $F_p(w_i)$ approaches to $F(w_i)$. In the case where the frequency of w_i is small, instead we use the frequency of its word class because low frequent data are statistically less reliable. We set the cutoff to 3.

The word classes used in computing the certainty score are the same as those used in protein name boundary expansion described in Figure 2 except for acronyms. As our preliminary experiment showed that more specific word classes for acronyms produced slightly better results on the reference corpus, we introduce new word classes (only for acronyms) as follows. First, given an acronym, capital letters, small letters, and numbers are converted to *A*, *a*, and *0*, respectively; and then, consecutive same characters are squeezed into one character; lastly, the last character is stripped if it is *0*. For instance, `HsMad1` will be first converted to *AaAaa0*, then squeezed into *AaAa0*, and *0* at the end is eliminated, resulting in a word class "*acronym_AaAa*". Table 2 shows some examples of acronym word classes and the words correspond to them.

Insert Table 2 about here

3.5 A Protein Name Dictionary

In addition to the previous steps described so far, we take advantage of a protein name dictionary compiled from the SWISS-PROT (Release 40.38) and TrEMBL (Release 22.6) protein databases (O'Donovan et al., 2002) in order to detect the protein names which are not recognized by the hand-crafted rules and/or a probabilistic model.

SWISS-PROT and TrEMBL contain 120,607 and 729,579 entries for proteins respectively as of December 2002. Figure 3 shows a fragment of the SWISS-PROT protein database (TrEMBL has the same format as SWISS-PROT).

Insert Figure 3 about here

We built a protein name dictionary by extracting protein names and their synonyms from the protein name fields (`DE` in Figure 3). In the example in Figure 3, "`14-3-3-like protein GF14 chi`" is a protein name, and "`General regulatory factor 1`" is its synonym. However, since the format of the `DE` field is sometimes inconsistent, it produces a number of incorrect protein names, leading to erroneous detection of protein names. Therefore, we excluded unlikely protein names which (a) begin with special characters (e.g., spaces, commas, and parentheses), (b) consist of only numerals or less than two characters, or (c) end with a hyphen or apostrophe.

In addition, we excluded the protein names composed of less than three tokens (words and symbols) since these protein names were found to harm the overall accuracy of identification. Furthermore, we applied the *detection rules* and *expansion rules* on the initial set of protein names

extracted from SWISS-PROT and TrEMBL, and those detected by the rules were discarded so as to improve processing speed by decreasing the dictionary size.

As a result, 114,876 of protein names and their synonyms were included in the dictionary.

3.6 Automatic Construction of Annotated Corpora

We proposed probabilistic models for expanding protein name boundaries and filtering out unlikely candidates, which require corpora annotated with protein names for training. However, creating annotated corpora needs biological expertise and is, even for human experts, time-consuming. Therefore, it is highly desirable to create a large-scale corpus in an automated way. Here, the important point to note is that training data do not need to be exhaustively annotated. That is, the total number of annotations is more important than coverage for training. Taking this into account, we propose a framework to automatically construct a large-scale corpus annotated with protein names.

The basic idea is that given a MEDLINE abstract we annotate protein names only if specific proteins are already known to be described in the abstract. These associations between proteins and MEDLINE abstracts can be easily extracted from a number of databases. In this study, we use the Protein Information Resource Non-redundant REFerence protein database (PIR-NREF) (Wu et al., 2002) containing 1,228,541 entries as of April 2003 (the release 1.21) due to its comprehensiveness. Figure 4 shows a fragment of the PIR-NREF protein database.

Insert Figure 4 about here

The example in Figure 4 indicates that the article associated with PubMed ID `10719003` is related to `maturase-like protein`. It does not ensure that the exact name `maturase-like protein` appears in the corresponding MEDLINE abstract because there are many variations of protein names. But if the protein name does appear in the text then we use the text as a source for

annotation. We extracted all pairs of protein names and PubMed IDs from the PIR-NREF database and annotated the protein names in cases where they appear in the corresponding MEDLINE abstracts. Figure 5 illustrates the process of automatic annotation.

Insert Figure 5 about here

As a result, we obtained a corpus containing 32,328 MEDLINE abstracts annotated with 91,773 occurrences of protein names. The corpus will be used for training the probabilistic models for protein name boundary expansion.

4 Evaluation

4.1 Overview

To evaluate the effectiveness of our method described above, we implemented a protein name identification system, named PROTEX, as illustrated in Figure 1 and conducted a series of experiments, in which our system was compared with the Yapex protein name identification system (Franzén et al., 2002; Olsson et al., 2002). There are three reasons this particular study was selected for comparison; according to our survey, Yapex is one of the state-of-the-art protein name identification systems based on hand-crafted rules; the system is publicly available through a CGI program at the *Proteinhalt i text* (protein concentration in text) project homepage (Franzén, 2003); and the annotated corpora used for the development and evaluation of Yapex are also publicly available.

The annotated corpora consist of a reference corpus and a test corpus, and they contain 99 and 101 MEDLINE abstracts, respectively. The reference corpus, which is annotated with 1,745 proteins, is used for training our probabilistic models for protein name boundary expansion and for computing the certainty scores, and the test corpus, which is annotated with 1,966 protein names,

is used for evaluation. Hereafter, we will call the reference corpus the *training set* and the test corpus the *test set*.

In the following experiments, we mainly compare four different models below for protein name identification. Notice that the protein name fragment detection step is done by heuristic rules in all cases (see Figure 1).

- *rule*: uses hand-crafted rules for protein name boundary expansion
- *rule+dic*: the same as *rule*, but also uses the protein name dictionary for identification
- *prob*: uses the probabilistic model for protein name boundary expansion and the filter (based on the certainty scores)
- *prob+dic*: the same as *prob*, but also uses the protein name dictionary for identification

4.2 Evaluation Metrics

Precision, recall, and F-score are used as evaluation metrics. Precision is the number of protein names a system correctly detected, divided by the total number of protein names detected by the system. Recall is the number of protein names a system correctly detected, divided by the total number of protein names contained in the input text. F-score combines these measures, recall and precision, into a single score and is defined as in Equation (8).

$$F\text{-}score = \frac{2 \times precision \times recall}{precision + recall} \tag{8}$$

For judgment of correctness, we follow Olsson et al. (2002) and use three criteria introduced in Yapex's evaluation: exact, partial, and fragment matches.[4] As for exact match, *every* fragment composing a protein name has to be correctly detected to be judged as correct, whereas, for partial match, a detected protein name is counted as correct in cases where *any* fragments composing the protein name are correctly detected. For fragment match, the counting unit is a fragment; that is, *each* fragment composing a protein name is to be judged independently whether it is correctly detected or not.

For instance, suppose that a protein name “ribosomal protein L22” is detected as “ribosomal **protein L22**,” where bold fonts indicate the detected fragments. In this case, it is not counted as an exact match because “ribosomal” was not detected, whereas it is counted as a partial match because at least one token composing the protein name were correctly detected. Lastly, it is counted as two fragment matches because two fragments out of three were correctly detected.

4.3 Results and Discussion

4.3.1 Overall Performance

Table 3 shows the result of the comparative experiment. The figures in the column “Yapex” are directly cited from the *Proteinhalt i text* project homepage (Franzén, 2003), and *rule*, *rule+dic*, *prob*, and *prob+dic* denote our proposed methods. A threshold for the certainty score (see Section 3.4) was set to 0.245 in this experiment, which was derived by applying two-fold cross-validation to the training data so as to maximize F-score for exact match. To put it more precisely, we divided the training data into two sets of text *A* and *B* in equal size, and used *A* for computing certainty scores for *B* and, in turn, used *B* for computing certainty scores for *A* with varying the threshold. Then we took an average of the thresholds which maximized F-score for each set.

Insert Table 3 about here

Comparing *rule* and *prob*, the latter produced better results in most cases. Especially, there is a marked increase in recall for fragment match from 66.5 to 75.6 (+12.0%), indicating that *prob* correctly expanded more protein name boundaries than *rule* (with a slight decrease of precision from 75.4 to 74.3 (−1.4%)). As a result, it led to higher precision in identifying exact protein names from 56.0 to 60.1 (+6.8%).

In cases where the protein name dictionary was consulted (i.e., *rule+dic* and *prob+dic*), the recall marginally improved by 0.8%–4.6% with the precision remaining steady irrespective of *rule* or *prob* used for protein name boundary expansion. Since dictionary lookup was done simply by exact match between dictionary entries and input in this study, more flexible string matching techniques may yield higher recall. Those techniques, such as one proposed by Krauthammer et al. (2001), should be explored in future work.

When compared to Yapex, our probabilistic methods, *prob* and *prob+dic*, produced lower precision irrespective of the criteria for judgment of correctness (2.0% to 8.4% decrease), while they showed higher recall than Yapex (2.4% to 13.0% increase) except for fragment match by *prob*. Consequently, their F-scores were found quite comparable to those of Yapex, despite the fact that our methods do not rely on POS taggers or syntactic parsers as used in Yapex.

We evaluated our methods on several criteria, i.e., exact, partial, fragment matches and recall, precision, and F-score. Which criterion is important depends on what purpose we use the system for. Considering our ultimate goal, i.e., text mining for the cancer-protein relations, exact match would be important for distinguishing numbers of protein names and for associating extracted information with them. As for recall and precision, high recall will be preferable in the case where comprehensive information is needed, while high precision will be preferable in the case where reliable information is needed. We will show later that higher precision can be achieved by restricting the system output based on the certainty score introduced in Section 3.4.

4.3.2 Performance for Compound Names

Since our proposed probabilistic model is focusing on name boundary expansion, it is expected to be more effective particularly for compound protein names (those composed of multiple tokens). To demonstrate the advantage, we evaluated Yapex and our method solely on compound protein names. The test set used in this evaluation is the same as the one used above and contains 897 occurrences of compound protein names. Table 4 shows the result. Since Yapex's performance for

compound names are not separately reported, it was obtained by submitting the test set to the Yapex demo page (Franzén, 2003) on March 27, 2003.

Insert Table 4 about here

In the case where only compound protein names are considered, *prob* mostly outperformed Yapex especially for exact match, indicating that it found protein name boundaries more accurately and more exhaustively than Yapex. This result verifies the advantage of our probabilistic model for finding compound protein names and their boundaries. Let us stress again that Yapex uses a syntactic parser to find name boundaries, whereas our method solely relies on surface clues without those NLP tools.

As for the rule-based method, *rule*, it results in poor performance in recall. This is due to the simplicity of our rule set for protein name expansion, and thus it could be improved by adding more comprehensive rules. However, Yapex's results suggest that it might result in a decline of precision unless carefully tuned; Yapex also adopts a rule-based method, showing higher recall but lower precision than *rule*.

The advantage of our method for compound names, at the same time, implies a disadvantage of our method for single-word names, since overall Yapex and our system produce comparable results (in terms of F-score). Table 5 provides the results regarding single-word names on the test set, where evaluation criteria for matching (i.e., exact, partial, and fragment matches) are not reported because there is no distinction among them for single-word names.

Insert Table 5 about here

Table 5 shows some decrease in F-score for our methods as compared with Yapex, but the difference (−2.9% for *prob*) was found not as significant as that for compound names. This is

mainly because the numbers of single-word and compound names are different in the test set, that is, since there are more single-word names, they have more influence on the overall performance.

In order to filter out unlikely single-word protein name candidates, Yapex makes use of a dictionary compiled from the SWISS-PROT database, which appears to contribute to improve the precision for single-word names. As discussed in Section 2, however, such a dictionary could be quickly obsolete as new protein names continue to be created. One possible solution is to continually update the dictionary. Yoshida et al. (2000) proposed a framework to automatically construct a protein name abbreviation dictionary from MEDLINE abstracts, which can be integrated into our system with a pre-compiled dictionary so as to improve the performance for single-word names.

4.3.3 Alternative Models

To deal with the data sparseness problem, we introduced a bigram class model for generalization, which is an integration of class-class and class-word transitional probabilities. To validate the effectiveness of the model, we compared it with two other alternatives: one without word classes (less generalized) and one without words (more generalized). Equation (9) and Equation (10) show the conditions of these models respectively for expanding name boundaries leftward. Notice that Equation (9) is the same as Equation (1) (i.e., before introducing a bigram class model).

$$P_p(w_{i-1}|w_i) \;>\; P_n(w_{i-1}|w_i) \tag{9}$$

$$P_p(c_{i-1}|c_i) \;>\; P_n(c_{i-1}|c_i) \tag{10}$$

Table 6 shows the results of protein name identification using our proposed model (*prob*) and the alternatives, in which "*word*" and "*class*" denote the word transition model defined as in Equation (9) and the class transition model as in Equation (10), respectively.

Insert Table 6 about here

In terms of partial match, there is less difference among these three models. This is expected by the definition of partial match: “detected protein names are judged as correct if any token composing them is correctly detected.” In other words, if any fragment of protein names is correctly detected by hand-crafted rules in the initial detection phase (which is in common for all models), it is regarded as correct. That is, name boundary expansion does not make much difference in performance for partial match. For other evaluation criteria, our proposed model outperformed the others, especially in exact match, indicating that our model is effectively generalized by combining transitional probabilities for words and classes.

4.3.4 The Use of an Automatically Constructed Corpus

We trained our proposed probabilistic model on the automatically constructed large-scale corpus, i.e., 32,328 MEDLINE articles annotated with 91,773 occurrences of protein names (see Section 3.6). Table 7 presents the results for overall performance, where “*manual*” and “*auto*” denote probabilistic models trained on manually annotated and automatically constructed corpora, respectively. Note that the results of *manual* are the same as those of *prob* shown in Table 3 and Table 4.

Insert Table 7 about here

Overall, the probabilistic model trained on the automatically annotated corpus (*auto*) achieved higher recall than the one based on manually annotated corpus (*manual*), indicating that *auto* found more protein names than *manual*, whereas the *manual* achieved higher precision than

auto, indicating that *auto* also produced false positives (i.e., detected as protein names but actually not). Consequently, *auto* favorably compares with *manual* in F-score, despite the fact that *auto* does not require human efforts to create annotated corpora.

The larger differences in the performance (recall and precision) for compound names are accounted for the automatically annotated corpus used for training *auto*. That is, the larger data set provided richer examples of protein names for training the name boundary expansion model (which resulted in higher recall), while it also provided more false examples (which resulted in lower precision). To improve the precision, more elaborate procedures need to be developed for creating higher quality training data. Figure 6 shows some examples of the resulting false positives (i.e., those protein names which were detected by our system but are not actually protein names according to the test set).

Insert Figure 6 about here

4.3.5 Filtering Based on the Certainty Scores

Lastly, the effectiveness of the certainty score introduced in Section 3.4 was examined. The certainty scores were computed using the training set (i.e., 99 MEDLINE abstracts). The system used here is *prob*; that is, the probabilistic models were used for expanding protein name boundaries. By varying a threshold for the certainty score, we drew a recall-precision curve in terms of exact match (Figure 7).

Insert Figure 7 about here

The right most and lowest circle corresponds to the result without restriction (i.e., threshold is 0). As threshold increased, precision gradually increased until recall fell to around 40%. Then

precision sharply increased up to around 90% with recall decreasing. In sum, although high precision was achieved, recall steeply dropped at the same time. To prevent recall from dipping, other features need to be explored for the certainty measure. For instance, surrounding words (contextual cues) may be effective.

Additionally, to investigate the validity of the filter on *actual* protein names, we applied it to 326,489 protein names extracted from SWISS-PROT and TrEMBL databases. (Note that, unlike the protein name dictionary construction procedure described in Section 3.5, short names and those detected by the *detection* and *expansion* rules were not excluded for this experiment.) Figure 8 gives the distribution of certainty scores computed for these protein names.

Insert Figure 8 about here

The distribution shows the peak between 0.25 and 0.30 and is positively skewed. According to the distribution, on the one hand, 207,016 names (63.4%) will successfully go through the filter in the case where the threshold is set to 0.245 (which was derived based on the training data and used in our experiments). On the other hand, 36.6% of actual protein names will be falsely discarded. We could recover those protein names by lowering the threshold, but it will also cause a decrease in precision as demonstrated in Figure 7. Once again, more effective features need to be investigated for the certainty measure.

5 Conclusions

In this paper, we presented a hybrid method for identifying protein names in biomedical texts with an emphasis on protein name boundary expansion. Our method utilizes a set of simple heuristics for initial detection of protein name fragments and takes advantage of a probabilistic model for expanding and finding protein name boundaries. The heuristics exploit surface clues reflecting the

characteristics of protein names, and the probabilistic model incorporates word classes for generalization so as to remedy the data sparseness problem. In addition, to detect those protein names which could not be identified by the rules and the probabilistic model, we complementarily utilized a protein name dictionary compiled from existing protein databases.

Our method, in contrast to the previous efforts, does not rely on POS taggers and/or syntactic parsers at all, since the information given by these NLP tools are not necessarily helpful for the task of protein name identification. This reduces both processing overhead and the potential number of probabilistic parameters to estimate. We implemented a protein name identification system, called PROTEX, based on our proposed method, and conducted comparative experiments to verify its effectiveness. The results demonstrated that PROTEX performed comparably to a best-of-the-breed system named Yapex which incorporates a syntactic parser. Moreover, in the case where only compound protein names were evaluated, our system outperformed Yapex in most cases. Furthermore, we proposed a framework for automatically constructing a large-scale corpus annotated with protein names for training a probabilistic model. Through an experiment, it was shown that the model trained on the automatically generated corpus favorably compares with one trained on the manually annotated corpus (in F-score). Lastly, we proposed a notion of certainty to filter out unlikely protein name candidates for improving precision; it was demonstrated to be effective to incrementally raise precision at the expense of recall.

6 Future Work

Future work would include a refinement of the certainty measure to prevent recall from rapidly falling. One of possible extensions is to utilize the probability estimates computed for name boundary expansion, so as to consider context (adjacent words) to some extent. In addition, integrating a framework of automatically discovering protein name abbreviations (for example, Yoshida et al., 2000) may improve the overall accuracy by filtering out unlikely single-word

protein name candidates.

References

Adamic, L. A., Wilkinson, D., Huberman, B. A., & Adar, E. (2002). A literature based method for identifying gene-disease connections. In *Proceedings of the IEEE computer society bioinformatics conference (CSB 2002)* (pp. 109–117).

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research, 25*(17), 3389–3402.

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., & Wheeler, D. L. (2003). Genbank. *Nucleic Acids Research, 31*(1), 23–27.

Brill, E. (1994). Some advances in rule-based part of speech tagging. In *Proceedings of the national conference for pattern recognition* (pp. 722–727).

Bruijn, B. de, & Martin, J. (2002). Literature mining in molecular biology. In *Proceedings of workshop on natural language processing in biomedical applications (NLPBA 2002)* (pp. 7–18).

Cercone, N., Lin, T. Y., & Wu, X. (Eds.). (2001). *Proceedings of the first IEEE international conference on data mining (ICDM 2001).* IEEE Computer Society Press.

Collier, N., Nobata, C., & Tsujii, J. (2000). Extracting the names of genes and gene products with a hidden Markov model. In *Proceedings of the 18th international conference on computational linguistics* (pp. 201–207).

Franzén, K. (2003). *Proteinhalt i text (protein concentration in text).* (Retrieved March 27, 2003, from `http://www.sics.se/humle/projects/prothalt/`)

Franzén, K., Eriksson, G., Olsson, F., Asker, L., & Lidén, P. (2002). Exploiting syntax when detecting protein names in text. In *Proceedings of workshop on natural language processing in biomedical applications (NLPBA 2002).*

Friedman, C., Kra, P., Yu, H., Krauthammer, M., & Rzhetsky, A. (2001). GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles. *Bioinformatics*, *17 Suppl. 1*, S74–S82.

Fu, Y., Mostafa, J., & Seki, K. (2003). Protein association discovery in biomedical literature. In *Proceedings of the third ACM/IEEE-CS joint conference on digital libraries* (pp. 113–115).

Fukuda, K., Tsunoda, T., Tamura, A., & Takagi, T. (1998). Toward information extraction: Identifying protein names from biological papers. In *Proceedings of the pacific symposium on biocomputing* (Vol. 3, pp. 705–716).

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, *40*, 237–264.

Grishman, R., & Sundheim, B. (1996). Message Understanding Conference - 6: A brief history. In *Proceedings of the 16th international conference on computational linguistics* (pp. 466–471).

Hirschman, L., & Blaschke, C. (2003). *BioCreAtIvE – critical assessment of information extraction systems in biology.* (Retrieved August 20, 2003, from `http://www.mitre.org/public/biocreative/`)

Hirschman, L., Park, J. C., Tsujii, J., Wong, L., & Wu, C. H. (2002). Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, *18*(12), 1553–1561.

Kazama, J., Makino, T., Ohta, Y., & Tsujii, J. (2002). Tuning support vector machines for biomedical named entity recognition. In *Proceedings of ACL workshop on natural language processing in the biomedical domain* (pp. 1–8).

Krauthammer, M., Rzhetsky, A., Morozov, P., & Friedman, C. (2001). Using BLAST for identifying gene and protein names in journal articles. *GENE*(259), 245–252.

Marsh, E., & Perzanowski, D. (1998). MUC-7 evaluation of IE technology: Overview of results. In *Proceedings of the 7th message understanding conference.*

Narayanaswamy, M., Ravikumar, K. E., & Shanker, V. K. (2003). A biological named entity recognizer. In *Proceedings of the pacific symposium on biocomputing* (Vol. 8, pp. 427–438).

Ng, S., & Wong, M. (1999). Toward routine automatic pathway discovery from on-line scientific text abstracts. In *Proceedings of genome informatics* (Vol. 10, pp. 104–112).

Nobata, C., Collier, N., & Tsujii, J. (1999). Automatic term identification and classification in biology texts. In *Proceedings of the 5th natural language processing pacific rim symposium* (pp. 369–374).

O'Donovan, C., Martin, M. J., Gattiker, A., Gasteiger, E., Bairoch, A., & Apweiler, R. (2002). High-quality protein knowledge resource: SWISS-PROT and TrEMBL. *Brief Bioinform, 3*(3), 275–284.

Ohta, T., Tateisi, Y., Kim, J., Mima, H., & Tsujii, J. (2002). The GENIA corpus: an annotated research abstract corpus in molecular biology domain. In *Proceedings of the human language technology conference* (pp. 73–77).

Olsson, F., Eriksson, G., Franzén, K., Asker, L., & Lidén, P. (2002). Notions of correctness when evaluating protein name taggers. In *Proceedings of the 19th international conference on computational linguistics* (pp. 765–771).

Palakal, M., Stephens, M., Mukhopadhyay, S., Raje, R., & Rhodes, S. (2002). A multi-level text mining method to extract biological relationships. In *Proceedings of the IEEE computer society bioinformatics conference (CSB 2002)* (pp. 97–108).

Proux, D., Rechenmann, F., & Julliard, L. (1998). Detecting gene symbols and names in biological texts: A first step toward pertinent information extraction. In *Proceedings of genome informatics* (Vol. 9, pp. 72–80).

Seki, K., & Mostafa, J. (2003). An approach to protein name extraction using heuristics and a dictionary. In *Proceedings of the american society for information science and technology annual conference (ASIST 2003)* (pp. 71–77).

Sekimizu, T., Park, H. S., & Tsujii, J. (1998). Identifying the interaction between genes and gene products based on frequently seen verbs in MEDLINE abstracts. In *Proceedings of genome informatics* (Vol. 9, pp. 62–71).

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

Thomas, J., Milward, D., Ouzounis, C., Pulman, S., & Carroll, M. (2000). Automatic extraction of protein interactions from scientific abstracts. In *Proceedings of the pacific symposium on biocomputing* (Vol. 5, pp. 538–549).

Witten, I. H., & Bell, T. C. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, *37*(4), 1085–1094.

Wu, C. H., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y., Hu, Z., Ledley, R. S., Lewis, K. C., Mewes, H., Orcutt, B. C., Suzek, B. E., Tsugita, A., Vinayaka, C. R., Yeh, L. L., Zhang, J., & Barker, W. C. (2002). The protein information resource: an integrated public resource of functional annotation of proteins. *Nucleic Acids Research*, *30*(1), 35–37.

Yoshida, M., Fukuda, K., & Takagi, T. (2000). PNAD-CSS: a workbench for constructing a protein name abbreviation dictionary. *Bioinformatics*, *16*(2), 169–175.

Zaki, M. J., Wang, J. T., & Toivonen, H. (Eds.). (2003). *Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining.* ACM Press.

Footnotes

[1]At the time of writing this paper on October 2003.

[2]The filter can also be applied in the case where rules are used for the name boundary expansion step, but we consider the filter as a part of a probabilistic framework described later and limit the use of the filter for an evaluation purpose.

[3]`http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html`

[4]Exact, partial, and fragment matches correspond to *strict*, *PNP*, and *sloppy*, respectively, proposed by Olsson et al. (2002).

Table 1

Examples of word classes.

Class	Examples
suffix_in	protein, oncoprotein, lactoferrin
suffix_ase	kinase, transferase, peptidase
word	the, a, an
acronym	CN, TrkA, USF
arabic_num	2, 3, 18, 76
roman_num	I, II, III, IV
roman_alpha	alpha, beta, gamma
punctuation	comma (,), period (.)
symbol	), (, %, +

Table 2

Examples of acronym word classes and some acronyms associated with them (not necessarily protein name fragments).

Acronym class	Examples
acronym_0A	90K, 3B, 32DC13
acronym_A0a	C1q, E3s
acronym_AaAa	HsMad1, HsRad52
acronym_aA0A	eIF4G

Table 3

A comparison between Yapex and our proposed methods on the test set of 101 MEDLINE abstracts. Numbers in parentheses indicate percentage increase/decrease relative to Yapex.

evaluation criteria		Yapex	*rule*	*rule+dic*	*prob*	*prob+dic*
exact	recall	59.9	65.0 (+8.5%)	66.1 (+10.4%)	66.9 (+11.7%)	67.7 (+13.0%)
	precision	62.0	56.0 (−9.7%)	56.3 (−9.2%)	60.1 (−3.1%)	60.2 (−2.9%)
	F-score	61.0	60.1 (−1.5%)	60.8 (−0.3%)	63.3 (+3.8%)	63.7 (+4.4%)
partial	recall	81.4	86.0 (+5.7%)	86.9 (+6.8%)	86.0 (+5.7%)	86.7 (+6.5%)
	precision	84.3	74.0 (−12.2%)	74.1 (−12.1%)	77.2 (−8.4%)	77.2 (−8.4%)
	F-score	82.8	79.6 (−3.9%)	80.0 (−3.4%)	81.4 (−1.7%)	81.6 (−1.4%)
fragment	recall	76.2	66.5 (−12.7%)	69.7 (−8.5%)	75.6 (−0.8%)	78.0 (+2.4%)
	precision	75.8	75.4 (−0.5%)	75.4 (−0.5%)	74.3 (−2.0%)	74.1 (−2.2%)
	F-score	76.0	70.7 (−7.0%)	72.5 (−4.6%)	75.0 (−1.3%)	76.0 (0%)

Table 4

A comparison between Yapex and our methods for compound protein names on the test set. Numbers in parentheses indicate percentage increase/decrease relative to Yapex.

evaluation criteria		Yapex	*rule*	*prob*
exact	recall	53.2	46.8 (−12.0%)	61.0 (+14.7%)
	precision	49.7	55.0 (+10.7%)	57.6 (+15.9%)
	F-score	51.4	50.6 (−1.6%)	59.3 (+15.3%)
partial	recall	73.3	63.4 (−13.5%)	78.6 (+7.2%)
	precision	68.5	74.6 (+8.9%)	74.3 (+8.4%)
	F-score	70.8	68.6 (−3.1%)	76.4 (+7.9%)
fragment	recall	65.8	53.0 (−19.5%)	69.8 (+6.1%)
	precision	65.1	75.0 (+15.2%)	66.4 (+2.0%)
	F-score	65.4	62.1 (−5.0%)	68.0 (+4.0%)

Table 5

A comparison between Yapex and our methods for single-word protein names on the test set (containing 1,069 single-word protein name occurrences). Numbers in parentheses indicate percentage increase/decrease relative to Yapex.

evaluation criteria	Yapex	*rule*	*prob*
recall	66.2	80.3 (+21.3%)	71.5 (+8.0%)
precision	71.4	56.4 (−21.0%)	62.6 (−12.3%)
F-score	68.7	66.3 (−3.5%)	66.7 (−2.9%)

Table 6

A comparison between our proposed model (prob) and word transition and class transition models on the test set (for all protein names). Numbers in parentheses indicate percentage increase/decrease relative to prob.

evaluation criteria		*prob*	*word*	*class*
exact	recall	66.9	41.8 (−37.5%)	41.6 (−37.8%)
	precision	60.1	41.0 (−31.8%)	48.0 (−20.1%)
	F-score	63.3	41.4 (−34.6%)	44.6 (−29.5%)
partial	recall	86.0	74.7 (−13.1%)	72.0 (−16.3%)
	precision	77.2	73.3 (−5.1%)	83.0 (+7.5%)
	F-score	81.4	74.0 (−9.1%)	77.1 (−5.3%)
fragment	recall	75.6	48.0 (−36.5%)	63.8 (−15.6%)
	precision	74.3	67.2 (−9.6%)	66.1 (−11.0%)
	F-score	75.0	56.0 (−25.3%)	64.9 (−13.5%)

Table 7

Results produced by probabilistic models trained on manually and automatically annotated corpora. Numbers in parentheses indicate percentage increase/decrease relative to manual.

evaluation criteria		compound		overall	
		manual	*auto*	*manual*	*auto*
exact	recall	61.0	66.3 (+8.7%)	66.9	68.4 (+2.2%)
	precision	57.6	51.6 (−10.4%)	60.1	57.3 (−4.7%)
	F-score	59.3	58.1 (−2.0%)	63.3	62.4 (−1.4%)
partial	recall	78.6	85.5 (+8.8%)	86.0	91.3 (+6.2%)
	precision	74.3	66.6 (−10.4%)	77.2	76.6 (−0.8%)
	F-score	76.4	74.9 (−2.0%)	81.4	83.3 (+2.3%)
fragment	recall	69.8	78.1 (+11.9%)	75.6	82.7 (+9.4%)
	precision	66.4	63.8 (−3.9%)	74.3	69.1 (−7.0%)
	F-score	68.0	70.2 (+3.2%)	75.0	75.3 (+0.4%)

Figure Captions

Figure 1. Framework of our proposed method for protein name identification.

Figure 2. The algorithm of word class assignment.

Figure 3. A fragment of the SWISS-PROT protein database.

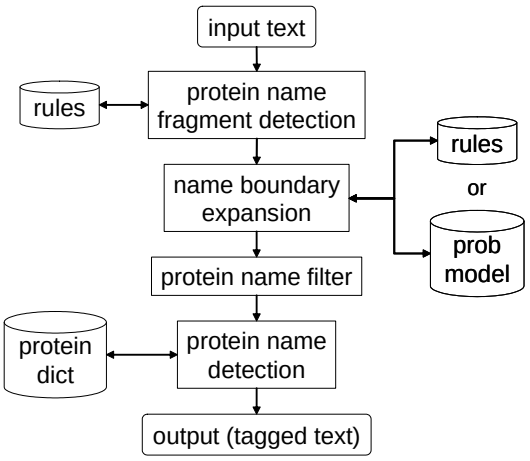
Figure 4. A fragment of the PIR-NREF protein database.

Figure 5. An illustration of automatic annotation for protein names.

Figure 6. Some examples of false positives produced by the probabilistic model trained on the automatically annotated corpus (*auto*).

Figure 7. The relation between recall and precision for exact match.

Figure 8. Distribution of the certainty scores computed for 326,489 protein names in SWISS-PROT and TrEMBL.


input text
rules
protein name fragment detection
rules
or
name boundary expansion
prob model
protein name filter
protein dict
protein name detection
output (tagged text)

variables

w: input word

s_w: suffix to be associated with w, which is a sequence of a vowel, one or more consonants (if any), and either a vowel or a consonant in the ending of w (can be `NULL` in case of no suffix)

c_w: output word class to be assigned to w

RA: a set of frequent Roman alphabets = {`alpha`, `beta`, `gamma`, `delta`, `epsilon`, `kappa`}

RN: a set of Roman numerals = {`I`, `II`, `III`, `IV`, $\cdots$}

PM: a set of punctuation marks = {comma (`,`), period (`.`), colon (`:`), semicolon (`;`)}

begin

if $w \in RA$ **then** c_w := *roman_alpha*

else if $w \in RN$ **then** c_w := *roman_num*

else if $w \in PM$ **then** c_w := *punctuation*

else if w is a number **then** c_w := *arabic_num*

else if w contains capital letter(s) **then** c_w := *acronym*

else if w is composed of alphabets **then**

  if s_w = `NULL` **then** c_w := *word*

  else if `length`(s_w) $\leq$ `length`(w)/2 **then** c_w := *suffix_*s_w

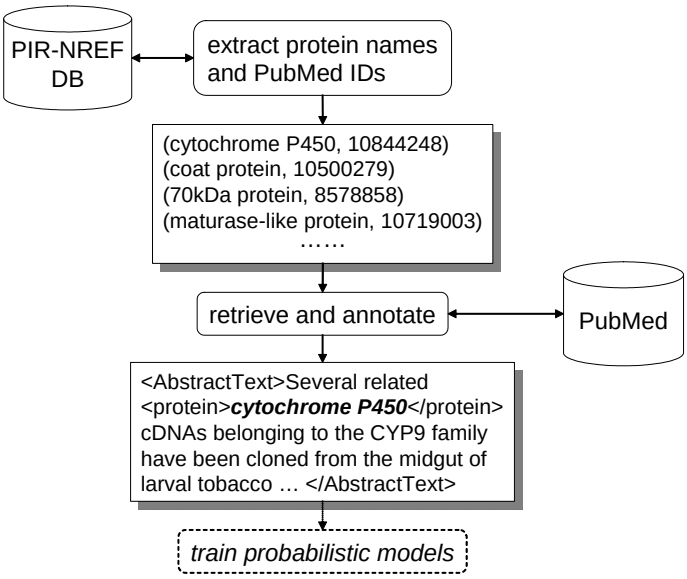
  else c_w := *word*

else c_w := *symbol*

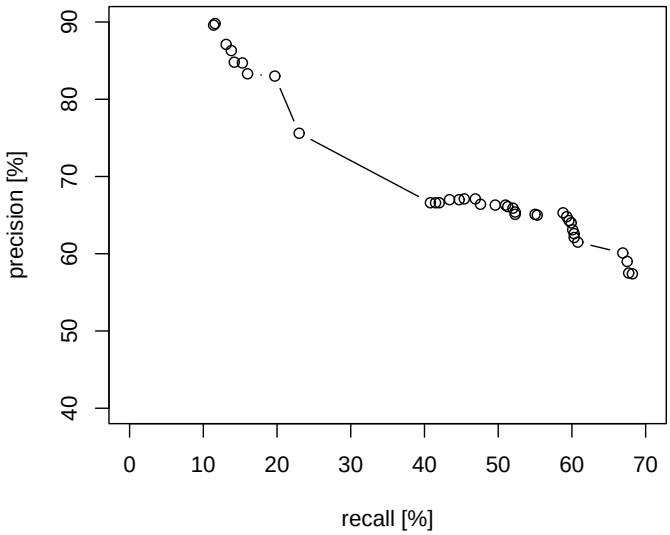
end

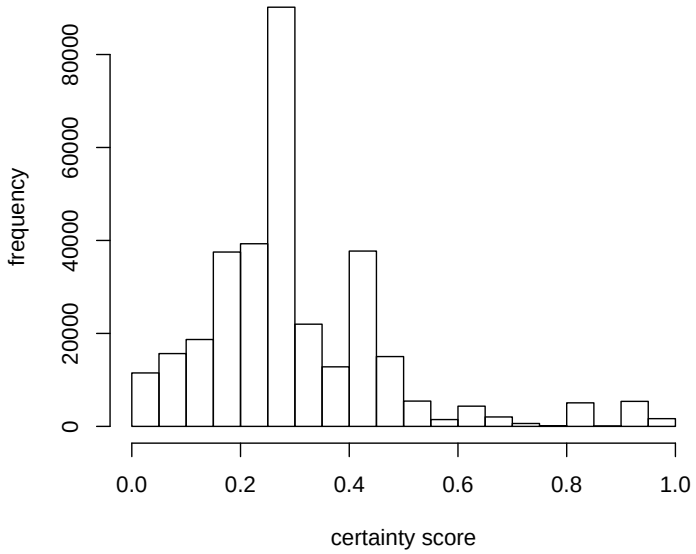
```
ID   1431_ARATH STANDARD; PRT; 267 AA.
AC   P42643; Q9M0S7;
DT   01-NOV-1995 (Rel.  32, Created)
DT   01-OCT-1996 (Rel.  34, Last sequence update)
DT   15-JUN-2002 (Rel.  41, Last annotation update)
DE   14-3-3-like protein GF14 chi (General regulatory factor 1).
GN   GRF1 OR AT4G09000 OR F23J3.30.
OS   Arabidopsis thaliana (Mouse-ear cress).
OC   Eukaryota; Viridiplantae; Streptophyta; Embryophyta;
     Tracheophyta;
```

```
<NrefEntry id="NF00000036" update_date="11-Mar-2002">
<protein_name>
maturase-like protein
</protein_name>
....
<bibliography>
<pmid>
10719003
</pmid>
</bibliography>
```


PIR-NREF DB
extract protein names and PubMed IDs
(cytochrome P450, 10844248)
(coat protein, 10500279)
(70kDa protein, 8578858)
(maturase-like protein, 10719003)
……
retrieve and annotate
PubMed
<AbstractText>Several related <protein>cytochrome P450</protein> cDNAs belonging to the CYP9 family have been cloned from the midgut of larval tobacco … </AbstractText>
train probabilistic models

80-kDa tyrosine	beta subunit
CD40L-treated B	copper cofactor
DNA-receptor	genistein suppress
HPP-CFCs	histidine-rich glycoprotein
K1K2-Xa	nuclear chromatin
MHC class II	perivascular DC migrate
RA time	receptor kinase
TPA-induced transcription	transcription regulatory factor


precision [%]
90
80
70
60
50
40
0
10
20
30
40
50
60
70
recall [%]


frequency
0
20000
40000
60000
80000
0.0
0.2
0.4
0.6
0.8
1.0
certainty score