Exploiting Syntax when Detecting Protein Names in Text

Gunnar Eriksson, Kristofer Franzén, Fredrik Olsson
Swedish Institute of Computer Science
Box 1263, SE-164 29 Kista, Sweden
{guer|franzen|fredriko}@sics.se

Lars Asker, Per Lidén
Virtual Genetics Laboratory AB
SE-171 77 Stockholm, Sweden
{lars.asker|per.liden}@vglab.com

Abstract

This paper presents work on a method to detect names of proteins in running text.

Our system — Yapex — uses a combination of lexical and syntactic knowledge, heuristic filters and a local dynamic dictionary. The syntactic information given by a general-purpose off-the-shelf parser supports the correct identification of the boundaries of protein names, and the local dynamic dictionary finds protein names in positions incompletely analysed by the parser.

We present the different steps involved in our approach to protein tagging, and show how combinations of them influence recall and precision.

We evaluate the system on a corpus of MEDLINE abstracts and compare it with the KeX system (Fukuda et al., 1998) along four different notions of correctness.

1 Background

The roles and functions of proteins are important study objects in many areas of the life sciences, as well as for the pharmaceutical industry. In view of the vast amount of scientific text produced in these areas, it would be useful to have methods for automatic structuring and extraction of information found therein.

The detection and categorization of named entities, such as names of people, organisations and places, in classical information extraction tasks such as in the Message Understanding Conferences (MUC) (Borthwick et al., 1998) might be regarded a solved problem. But names of proteins present a slightly different challenge because of their variant structural characteristics and the specifics of the text domains in which they appear. This certainly holds true for other biological substances, and probably for many other kinds of named terminology as well.

One common reason for developing methods for automatic detection of protein names in text has been the desire to build systems for automatic extraction of interactions between proteins (Blaschke et al., 1999; Thomas et al., 2000). However, the detection of protein names is in itself useful. In our case, the first application at hand is a browsing support system, which links protein names in scientific text to entries in SWISS-PROT (Bairoch and Apweiler, 2000). Since the intended user is likely to be a domain expert able to judge if a hyperlink actually refers to a protein, some false links might be acceptable. On the other hand, too many words erroneously marked as proteins, would give the user sore eyes and little confidence in the system.

Previous attempts at identifying protein names in text can be divided into systems using machine learning techniques (Nobata et al., 1999; Collier et al., 2000) and systems based on hand-written rules (Fukuda et al., 1998; Humphreys et al., 2000). The advantage of using machine learning techniques is that such a system is relatively easy to tune to new domains, provided that tagged training data exist. A rule-based system, on the other hand, requires a lot of human analysis and labour, but results in a transparent system which is easier to support, adjust and expand. The system described and evaluated in this paper — Yapex — is based on hand-written rules and utilises information from an off-the-shelf syntactic parser to improve performance.

Work on evaluation of protein name taggers seldom clearly specify what notion of correctness has been used when evaluating the systems, with the exception of de Bruijn and Martin (2000), who present figures on *undertagging* and *overtagging*, as well as *type* and *token* matches.

In this work we introduce the four different notions of correctness that we have used when evaluating the system. The different notions of correctness stress different characteristics of Yapex and the KeX system (Fukuda et al., 1998) with which we compare it.

Correspondingly, the definition of what should be regarded a protein name is often implicit in previous work in this area. In the following section we describe what we consider to be a protein name.

In section 3 we describe the algorithm of the Yapex-system and in section 4 we give an account of the evaluation of the system and a comparison with the KeX system. Finally, we discuss the results in section 5.

2 Protein Names

Despite the lack of common standards and fixed nomenclatures, protein names exhibit several regularities that can be exploited in order to identify previously unseen instances. Primarily, protein names are almost always descriptive in some way. Protein characteristics such as function (e.g., *growth hormone*), localization or cellular origin (such as *HIV-1 envelope glycoprotein gp120*), physical properties (*salivary acidic protein-1*), similarities to other proteins (*Rho-like protein*) are commonly reflected in the name. Names are also constructed using a combination or abbreviation of the above.

It needs to be said that the definition of what should be considered as a protein name is not self-evident and that it can be varied to a certain extent. In this study we define a protein as a single biological entity composed of one or more amino acid chains. Protein fragments or protein families are not included in this definition. Furthermore, since names of genes and the names of their protein products are used equivocally we make no attempt to distinguish between them.

In addition to the semantic definition above, from a text structural point of view, we define a protein name as a sequence of words denoting a specific, individual protein entity. Furthermore, we also include some, more indirect, references to individual protein entities into the protein name definition, (e.g. `<protein>`*importin beta1*`</protein>` *derivatives*). The definition excludes non-specific reference to individuals (*transcription factor*, *a 89 kD protein*). It also excludes most reference to groups or classes of proteins (*protein kinases*, *globulins*), though phrases denoting small groups of nearly identical proteins are included (*eukaryotic RhoA-binding kinases*). Finally, the definition excludes anaphoric (intra-textual) references to proteins (*these proteins*).

3 Method

Arguably, building information extraction systems always involves decisions regarding how to balance recall and precision; depending on the application, one may want to focus on one or the other. Yapex initially strives for high recall with the consequence of poor precision. Later modules in the pipelined system use filtering techniques and syntactic information to boost precision, and a local dynamic dictionary is eventually applied to increase recall.

The Yapex algorithm can be described as consisting of the seven steps described below: The first four steps are concerned with the lexical analysis of single word tokens, and the first two of these are implementations of some of the heuristic steps in the algorithm described by Fukuda et al. (1998) from which the terminology of these steps is borrowed. Steps five and six are concerned with the syntactic analysis of noun phrases and of the lexical categories derived in the previous steps, and the final step utilizes the syntactic information gathered to identify new single- or multi-word protein names.

3.1 Lexical analysis of feature terms

Feature terms are words, e.g., *receptor* and *enzyme*, that describe the function or characteristics of a protein. These words often occur in or nearby a protein name and can be used as indicators of the presence of such a name. The analysis discriminates between internal and external feature terms, internal terms being words that belong to the name like *protein*, *particle*, and *receptor*. External feature terms are words — e.g., *peptide*, *domain*, and *terminal* — that act as indicators of a protein name but, most often, do not constitute a part of the name itself, according to our protein name definition. Among the internal feature terms we treat strong terms separately. These terms (*factor*, *receptor*, and *enzyme*) are even stronger indicators of a protein name. We currently tag words as feature terms if we find them in our list of about 50 such words.

3.2 Lexical analysis of core terms

A core term constitutes the nucleus of a protein name. These terms are the parts of a protein name that show the closest resemblance to regular proper names in that the principles for their coining vary, and often are rather arbitrary. As candidates for these terms we pick words ending in *-ase* and *-in*, or strings with characteristics typical of protein names, i.e., strings containing instances of upper case letters or numbers, found in names of proteins like *HsMad2* and *U3-55k*. Furthermore, as all protein names do not conform to the patterns above, words are dubbed core terms if they are found in a list of established protein names such as *interferon*.

Two general filters are applied to these terms to avoid overgeneration: Words consisting of $\geq$ 50% non-word characters, and measuring units are discarded as core terms.

3.3 Lexical analysis of specifiers

Yapex also recognizes a third lexical category, the specifier. Specifiers are terms that often occur in the beginning or end of a protein name to, e.g., specify an individual protein. We treat Arabic and Roman numerals, letters, Greek letter names, and combinations of these as specifiers.

3.4 Applying filters and knowledge bases

To remedy the low precision obtained in the previous step, a set of filters is applied to get rid of false hits. Some filters use regular expression patterns of word suffixes to rule out, e.g., names of chemical substances. Other filters use patterns of whole words/expressions to filter out, e.g., personal names and other parts in bibliographical references, chemical formulas, arithmetic expressions, and amino acid sequences. A third group of pattern-matching filters remove the core term annotation on words unlikely to function as core terms: words $\geq$ 6 characters long consisting solely of upper case letters, or consisting of upper case letters and more than one hyphen are discarded.

Short core terms ($\leq$ 3 characters) get special treatment. Only those found in our short-protein-name knowledge base drawn from SWISS-PROT are considered core terms. All the others are tagged as potential core terms to be used later in the protein name identification process. Core terms resembling regular proper names are treated the same way.

3.5 Finding noun phrases

In order to enhance detection of name boundaries, this step takes advantage of the Functional Dependency Grammar (FDG) parser from Conexor Oy (Tapanainen and Järvinen, 1997), which produces full syntactic analysis with information about dependencies and phrasal heads. For every noun phrase, we identify the head and its preceding lexical modifiers. This constitutes the minimal noun phrase — the noun phrase without any subordinate noun phrases — and is considered a potential protein name location.

3.6 Identifying protein names

To identify the protein name we start by adjoining all specifiers to their preceding core, potential core, or feature term. Then all external or plural feature terms, their adjoined specifiers, and words without a lexical analysis from Yapex are stripped off from the right edge of the noun phrase. From the left edge, words earlier identified as numerals together with measuring units are stripped off. The remaining part of the noun phrase is considered a potential protein name. It is selected as such if it contains a core term, a strong feature term together with at least one other word token, a feature term with an adjoined specifier, or a potential core term together with a feature term somewhere in the unstripped noun phrase.

3.7 Applying a local dynamic dictionary

The relevant terms of the protein names identified in the previous step are stored in a local dictionary as regular expressions. For every document, this dictionary is used in an additional tagging pass over the text, so that protein names that already have been found, can be flexibly matched to protein names enclosed in noun phrases undetected or misinterpreted by the parser.

4 Evaluation

At this point we can present results of our system (Yapex) applied to a corpus of 99 MED-

LINE abstracts containing 1745 protein names tagged by domain experts.

The first aim of the current evaluation, which is performed on data also used for reference during development, is to see how much each combination of the steps described in 3.4 and 3.7 contributes to the final result.

All four cases described below include the same way of tagging feature terms and core terms, employing the FDG parser to find minimal noun phrases, and mechanisms for identifying protein names.

YAPEX	NO LDD	LDD
NO FKB	$R = 88.0\%$	$R = 97.2\%$
	$P = 62.4\%$	$P = 61.0\%$
	$F = 73.1\%$	$F = 75.0\%$
FKB	$R = 78.7\%$	$R = 88.6\%$
	$P = 80.8\%$	$P = 79.6\%$
	$F = 79.8\%$	$F = 83.8\%$

Table 1: Results varying along Local Dynamic Dictionary (LDD) and Filters and Knowledge Bases (FKB), given in recall (R), precision (P), and F-score (F) under the SLOPPY condition (see below).

The motivation for using a local dynamic dictionary is to increase recall. Contrary to our intuition, Table 1 illustrates that precision did not seem to drop severely even though recall increased with 10.5% and 12.6% (from 88.0% to 97.2% and from 78.7% to 88.6%) when toggling the use of a local dynamic dictionary as regards the use of external filters and knowledge bases.

The second aim of the evaluation is to investigate how the use of syntactic information, i.e., the use of the syntactic parser information as described in sections 3.5–3.7, influences our results. To compare our approach with a system that reports good results without the explicit use of syntax, we use the KeX system as a baseline. KeX[1] is a freely available protein name annotation tool based on the algorithms presented in Fukuda et al. (1998).

In Table 2, Yapex and KeX are compared in terms of precision, recall and F-score[2] when evaluating the performance under four different conditions of correct matching:

SLOPPY: If any part of the proposed hit matches some part of the answer key, the hit is counted as a match.

PROTEIN NAME PARTS (PNP): Any part of the hit that matches any part of the answer key is counted as one match. This is a quantification of the SLOPPY match that gives the degree of overlap between the proposed hit and the answer key.

STRICT: If a proposed hit matches one answer key exactly, the hit is counted as a match.

LEFT OR RIGHT BOUNDARY: If a proposed hit exactly matches any boundary of the answer key, the hit is counted as a match.

	YAPEX	KEX
SLOPPY	$R = 88.6\%$	$R = 82.4\%$
	$P = 79.6\%$	$P = 72.3\%$
	$F = 83.8\%$	$F = 77.0\%$
PNP	$R = 78.2\%$	$R = 68.4\%$
	$P = 66.1\%$	$P = 39.8\%$
	$F = 71.7\%$	$F = 50.3\%$
STRICT	$R = 67.8\%$	$R = 39.3\%$
	$P = 60.9\%$	$P = 34.5\%$
	$F = 64.2\%$	$F = 36.8\%$
LEFT	$R = 77.3\%$	$R = 54.7\%$
OR	$P = 69.4\%$	$P = 48.0\%$
RIGHT	$F = 73.2\%$	$F = 51.1\%$

Table 2: Results for Yapex and KeX given in recall (R), precision (P), and F-score (F).

The first thing to notice from Table 2 is that under the SLOPPY condition the two systems perform on a comparable level. Yapex performs slightly better, but this difference could be due to lack of KeX training on this corpus, or a difference in the definitions of what constitutes a protein name. We notice though that it is only under this condition that KeX performs close to the results reported in de Bruijn and Martin (2000), but not at all close to what is reported in Fukuda et al. (1998).

Visualizing the F-scores in Figure 1, it is clear that both a STRICT and a PNP definition of

[1] KeX can be downloaded from http://www.hgc.ims.u-tokyo.ac.jp/service/tooldoc/KeX/intro.html.

[2] F-score is a measure combining precision and recall:

$$F = \frac{(\beta^2+1)PR}{(\beta^2 P + R)}$$

where β is a parameter that represents the relative importance of Precision (P) and Recall (R), in our case equally important (i.e., $\beta = 1$).

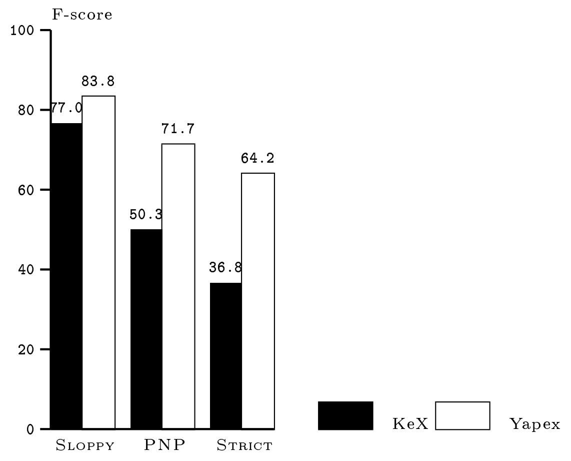

Figure 1: F-score for Yapex and KeX when evaluated on our corpus.

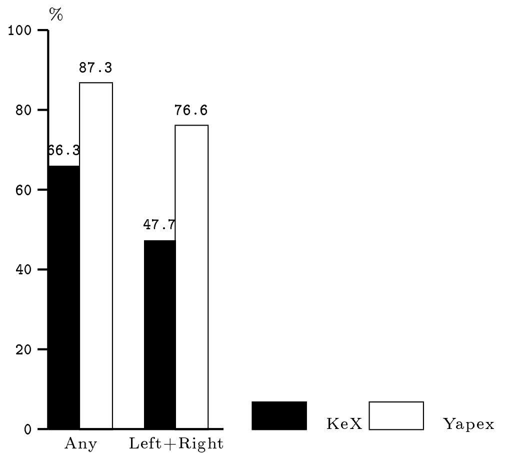

Figure 2: Given a SLOPPY hit, this chart shows the probability of finding protein name boundaries for Yapex and KeX.

a match favour the Yapex system. The result under the PNP condition clearly shows that the overlap between the proposed hits and the corresponding answer keys is remarkably higher for Yapex than for KeX, i.e., when the protein names consist of more than one word Yapex will find more of these name parts. We believe that this is due to the ability of the parser to analyse noun phrases, and thereby predict the boundaries of protein names.

When looking at the result under the STRICT condition, the impression remains the same, suggesting that Yapex is much better at finding the exact edges of the protein names. This is also shown by the result under the LEFT OR RIGHT condition in the last row of Table 2. In fact, this difference is further emphasized if we look at only the correct hits under the SLOPPY condition. Looking at the result this way (Figure 2), we find that Yapex recognizes the correct boundaries in 76.6% of all cases and identifies any of the boundaries correctly at a rate of 87.3%. The corresponding figures for KeX is 47.7% and 66.3%.

5 Discussion

Tagging of protein names in running text is cumbersome even for human domain experts, and evaluation of a protein name tagger requires a tagged corpus. Even though there exists a publicly available corpus of tagged MEDLINE abstracts developed in the GENIA project (Collier et al., 1999), we have chosen to evaluate the system on our reference corpus tagged by domain experts, since it turned out that our definition of protein names was not fully compatible with the subclasses of the GENIA protein ontology. Soon, we will be able to present results from running the systems on a separate test corpus. For an exhaustive discussion on the problems of building annotated corpora for the molecular-biology domain, and results on inter-annotator agreement, cf., Tateisi et al. (2000).

To problematize the metrics of recall and precision, we have chosen to evaluate along several notions of correctness. What is relevant to annotate varies with the intended application, and different methods of evaluation can highlight characteristics of competing systems. PNP is a relevant measure for this kind of named terminology where even human domain experts argue about the boundaries of names, since it gives an idea of how much of the multi-word proteins the systems match.

We have shown that a system without elaborate syntax performs weaker than our system in detecting protein names, with respect to boundaries as well as content. There is nothing surprising about a syntactic parser being able to aid in the detection of protein names; names cannot be found anywhere but in noun phrases. Given a perfect parser that identifies minimal noun phrases, the problem would be reduced to deciding if the noun phrase is a protein name or not. It should be noted though, that we use the FDG parser without modification; it has not been trained to handle this quite specific subdomain of text. Our technique of boosting the identification of noun phrases by the Local Dynamic Dictionary finds noun phrases that where not correctly analysed as such by the parser.

We believe that the syntactic information given by the parser is not only of use for protein name detection, but will also be of considerable help in forthcoming work in analysing the relations in which the detected proteins participate.

Acknowledgements

Partial funding for this project has been provided by VINNOVA, the Swedish Agency for Innovation Systems.

References

Amos Bairoch and Rolf Apweiler. 2000. The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucl. Acids. Res.*, 28:45–48.

Christian Blaschke, Miguel A. Andrade, Christos Ouzounis, and Alfonso Valencia. 1999. Automatic extraction of biological information from scientific text: protein—protein interactions. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 60–67, Heidelberg, Germany, August 6-10.

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. NYU: Description of the MENE Named Entity System as used in MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Fairfax, VA, USA, April 29 - May 1.

Nigel Collier, Hyun Seok Park, Norihiro Ogata, Yuka Tateishi, Chikashi Nobata, Tomoko Ohta, Tateshi Sekimizu, Hisao Imai, Katsutoshi Ibushi, and Jun ichi Tsujii. 1999. The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers. In *Proceedings of the European Association for Computational Linguistics (EACL) conference.*

Nigel Collier, Chikashi Nobata, and Jun ichi Tsujii. 2000. Extracting the Names of Genes and Gene Products with a Hidden Markov Model. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 201–207, August.

Berry de Bruijn and Joel Martin. 2000. Protein Name Tagging. Presented as a poster at the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB'00).

Ken-ichiro Fukuda, Tatsuhiko Tsunoda, Ayuchi Tamura, and Toshihisa Takagi. 1998. Toward Information Extraction: Identifying Protein Names from Biological Papers. In *Proceedings of the Pacific Symposium on Biocomputing (PSB'98)*, pages 705–716, Maui, Hawaii, January 4-9.

Kevin Humphreys, George Demetriou, and Robert Gaizauskas. 2000. Two Applications of Information Extraction to Biological Science Journal Articles: Enzyme Interactions and Protein Structures. In *Proceedings of the 5th Pacific Symposium of Biocomputing*, pages 72–80.

Chikashi Nobata, Nigel Collier, and Jun ichi Tsujii. 1999. Automatic Term Identification and Classification in Biology Texts. In *Proceedings of the Natural Language Pacific Rim Symposium (NLPRS'2000)*, pages 369–374, November.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71, Washington D.C., April. Association for Computational Linguistics.

Yuka Tateisi, Tomoka Ohta, Nigel Collier, Chikashi Nobata, and Jun ichi Tsujii. 2000. Building an Annotated Corpus in the Molecular-Biology Domain. In *Proceedings of Workshop on Semantic Annotation and Intelligent Content*, Centre Universitaire, Luxembourg, August. ACL. The workshop was held in conjuntion with the 18th International Conference on Computational Linguistics (COLING-2000).

James Thomas, David Milward, Chirtos Ouzounis, Stephen Pulman, and Mark Carroll. 2000. Automatic Extraction of Protein Interactions from Scientific Abstracts. In *Proceedings of the Pacific Symposium on Biocomputing (PSB 2000)*, pages 538–549, Oahu, Hawaii, January 4-9.