Machine Learning for Protein Name Extraction

Serhan TATAR
Bilkent University
statar@bilkent.edu.tr

Abstract
A large volume of protein data has been generated as a result of biological research. This vast amount of data is generally stored in the textual form in databases such as Medline. Currently, over 11 million summaries of articles are stored in Medline. However, lack of formal structure in the articles makes difficult to retrieve and process the information stored in these articles. In this paper, we explore the use of machine learning techniques for the information extraction task and present the initial results of the conducted experiments. Particularly, we study using Hidden Markov Models (HMMs) for protein name extraction from the biological texts.

Keywords
Information Extraction, Machine Learning, Hidden Markov Models, Protein Name Extraction.

INTRODUCTION
Biological knowledge, generated as a result of biological research in the past, is currently stored in published articles in scientific publications. Medline, a rich knowledge source for biological information, currently contains over 11 million abstracts of these articles. Although Medline contains enormous amount of biological information, there is no formal structure in which Medline can present the stored information. Stored information is generally found in the textual form. Lack of formal structure in the articles makes it tedious and time-consuming to retrieve and process the information stored in these articles. There is a growing demand for automatic discovery and extraction of information from biological texts. Information extraction systems can be used to meet the need for automatic discovery and extraction of information from biological texts.

Information Extraction (IE) may be defined as the task of extracting relevant information from a specific document set. As a discipline, information extraction is as old as the Message Understanding Conference (MUC) (Def, 1995), the forum that defined the problem. MUC-style IE problem can be defined as the identification of instances of a particular class of events or relationships in a natural language text, and the extraction of the relevant arguments of the event or relationship. IE has figured prominently in the field of NLP. However, its domain-specific nature makes it difficult to adapt an IE system to a new domain. In order to overcome this adaptation problem, many researches address using machine learning algorithms to perform IE tasks [3].

In this paper, we explore the use of machine learning techniques to extract protein names from the biological text and present the initial results of the conducted experiments. The structure of the paper is as follows. Section 2 describes the previous work, which has been done until now on the subject. After reviewing the previous work, we present the example corpora used in the experiments in Section 3. In Section 4, protein name extraction problem and the implemented machine learning approach are defined. Section 5 describes the experimental evaluation of the study. Finally, in the last section we discuss our conclusions, pointing towards future research.

RELATED WORK
Many research projects [5, 6] have focused on the manual development of IE systems for the discovery and extraction of information from biological texts. As stated above, manual adaptation of IE systems to specific domains is tedious and time-consuming. Moreover, human error during rule generation makes these systems error-prone.

Recent research on the subject has shown that machine learning techniques can be used in order to perform protein name extraction from texts. Moreover, many recent projects [1, 2, 8, 9, 10, 11, 12] have focused on the automatic extraction of protein names from biological texts using machine learning methods.

Hidden Markov Models are among the successful statistical learning techniques. Moreover, recent research [8, 10] in the IE community has shown that HMMs can be of service, both in performing fragment-to-slot mapping and in solving associated tasks.

In this study, we applied HMMs to protein name extraction task. We combined various approaches on the subject, made enhancements to get better performance, and present the results of the conducted experiments

EXAMPLE CORPORA

In order to conduct experiments, we use YAPEX [7] corpora, which contains two collections that consist of Medline abstracts, obtained in the following way.

A document set was obtained by posing the query "protein binding [Mesh term] AND interaction AND molecular" with the parameters "abstract", "English", "human", and "publication date 1996-2001" to Medline. From this set 99 abstracts were drawn randomly to form the reference (training) collection. Another non-overlapping set of 48 abstracts was drawn to form a part of the test collection.

The remaining 53 abstracts of the 101 in the test collection correspond to a randomly chosen, re-tagged subset of the GENIA corpus [4] containing 723 annotated protein names.

The reference and test corpora are mutually exclusive. The corpora are available for download at "http://www.sics.se/humle/projects/prothalt/". Sample tagged data is shown in Figure-1.

<PubmedArticle>
<MedlineID>21294781**</MedlineID>**
<PMID>11401507**</PMID>**
<ArticleTitle>Molecular dissection of the **<Protname>**importin beta1**</Protname>**-recognized nuclear targeting signal of **<Protname>**parathyroid hormone-related protein**</Protname>**.**</ArticleTitle>**
<AbstractText>Produced by various types of solid tumors, **<Protname>**parathyroid hormone-related protein**</Protname>** (**<Protname>**PTHrP**</Protname>**) is the causative agent of humoral hypercalcemia of malignancy. The similarity of **<Protname>**PTHrP's**</Protname>** amino-terminus to that of **<Protname>**parathyroid hormone**</Protname>** enables it to share some of the latter's signalling properties, but its carboxy-terminus confers distinct functions including a role in the nucleus/nucleolus in reducing apoptosis and enhancing cell proliferation. **<Protname>**PTHrP**</Protname>** nuclear import occurs via a novel **<Protname>**importin beta1**</Protname>**-mediated pathway. The present study uses several different direct binding assays to map the interaction of **<Protname>**PTHrP**</Protname>** with **<Protname>**importin beta**</Protname>** using a series of alanine mutated **<Protname>**PTHrP**</Protname>** peptides and truncated human **<Protname>**importin beta1**</Protname>** derivatives. Our results indicate that **<Protname>**PTHrP**</Protname>** amino acids 83-93 (KTPGKKKKGK) are absolutely essential for **<Protname>**importin beta1**</Protname>** recognition with residues 71-82 (TNKVETYKEQPL) additionally required for high affinity binding; residues 380-643 of **<Protname>**importin beta1**</Protname>** are required for the interaction. Binding of **<Protname>**importin beta1**</Protname>** to **<Protname>**PTHrP**</Protname>** is reduced in the presence of the GTP-bound but not GDP-bound form of the guanine nucleotide binding protein **<Protname>**Ran**</Protname>**, consistent with the idea that **<Protname>**Ran**</Protname>**GTP binding to **<Protname>**importin beta**</Protname>** is involved in the release of **<Protname>**PTHrP**</Protname>** into the nucleus following translocation across the nuclear envelope. This study represents the first detailed examination of a modular, non-arginine-rich **<Protname>**importin beta1**</Protname>**-recognized nuclear targeting signal. Copyright 2001 Academic Press.**</AbstractText>**
</PubmedArticle>

Figure 1: Sample Tagged Medline Abstract

In the corpora each article has four parts:

- *MedlineID*: starts with <MedlineID> tag and ends with </MedlineID>.
- *PMID*: starts with <PMID> tag and ends with </PMID>.
- *ArticleTitle*: starts with <ArticleTitle> tag and ends with </ArticleTitle>.
- *AbstractText*: starts with <AbstractText> tag and ends with </AbstractText>.

Last two parts, ArticleTitle and AbstractText, contain protein names. In the figure, tagged protein names can be seen clearly. Each protein name is marked by two tags: <Protname> and </Protname> (e.g. <Protname> retinoic acid receptor alpha </Protname>).

PROTEIN NAME EXTRACTION

Problem Definition

The task of extracting protein names from biomedical corpora is still a challenge, due to the following reasons:

- Many instances of new protein names do not suit exactly the standard terminology.
- Authors often refer to proteins already stored in protein databases using variations, which do not exist in the databases.

The success of a protein name extraction method depends on how well it recognizes the regularities of protein naming and name variations. Our approach was to start with a set of protein names collected from the training set, and then extend it using a carefully designed method. We centered our effort around this initial set of proteins, our aim being to develop a highly accurate information extraction system. The major task was to generalize the coverage of the training set, while at the same time trying to minimize any decrease in accuracy.

Representing Fragments & Generalization

Despite the lack of common standards and irregularities, and all the problems stated above, protein names exhibit several regularities that can be exploited in order to perform generalization [7]. First of all, protein names are almost always depictive. 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. As can be noted from the examples, protein names often consist of multiple words.

As stated above, the primary task of the learner is to generalize the coverage of the training set. Therefore, our first aim is to design a method that provides both generalization and accuracy. Generalizing means to recognize the parts susceptible of being changed in new protein names, and replace them with generic placeholders [1]. Thus, following the approach used in [8], we generalize protein names by using words and word-type information. We determine eight types for the words:

- *SingleLetter:* Single-letter words are in this type.
- *Number:* Numbers are in this type.
- *RomanNumeral:* Roman numerals are in this type.
- *GreekLetter:* Many protein names contain Greek letters (e.g. retinoic acid receptor alpha). We group Greek letters into this type.
- *Abbrv:* Abbreviations are in this type.
- *Regular:* Regular words are in this type.
- *Delimeter:* Delimiters are in this type.
- *Unknown:* This type contains the words that can not be grouped into the above types.

Table-1 shows word-type patterns and some examples.

Patterns	
SingleLetter:	[a-zA-Z]
Number:	[0-9]+
RomanNumeral:	I|II|III|IV|V|VI|VII|VIII|IX|X|XI|XII|XIII|XIV|XV|XVI|XVII|XVIII
GreekLetter:	alpha|beta|gamma|delta|epsilon|theta|kappa|lambda|sigma|mu
Abbrv:	[a-zA-Z]+)(([A-Z][a-z]*)|([0-9]+))(([a-zA-Z]+)|([0-9]+)|['])*
Regular:	[a-zA-Z][a-z']+
Delimeter:	[.,;:(){}/]|-
Unknown:	

Examples	
SingleLetter:	a
Number:	768
RomanNumeral	I
GreekLetter:	alpha
Abbrv:	AB
Regular:	Abnormal
Delimeter:	(
Unknown:	80%

Table 1: Word-type patterns and examples

IE and Hidden Markov Models (HMMs)

As compared to many other techniques used in natural language processing, HMMs are an extremely flexible tool and has been successfully applied to a wide variety of stochastic modeling tasks.

Hidden Markov Models are the stochastic analogs of finite state automata. A stochastic FSA is a generalization of a deterministic FSA in which each transition and each accepting state has an associated probability. Associated emission probabilities define the likelihood of a state to emit various tokens. The transitions from a given state have an associated transition distribution which defines the likelihood of the next state given the current state. For any given state in such an automaton, the probability of acceptance (i.e., of the state being terminal) and the probabilities of its outgoing transitions must all sum to one. Thus, a probability can be associated with any sequence belonging to the language the FSA models. This membership probability is the product of the transition probabilities along the unique state trajectory encoded by the sequence, and the acceptance probability of the terminal state.

HMMs and IE are widely discussed in [10]. Formally, HMMs are composed of a set of states Q, with specified initial and final states q_1 and q_F, a set of transitions between states ($q \rightarrow q'$), and output symbols $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$. The model generates a string $X=x_1x_2\ldots x_l$ by beginning in the initial state, following the allowed possible states and reaching the final state. We denote one state follows another by $P(q \rightarrow q')$ and a state emits a particular output symbol by $P(q \uparrow \sigma)$. Finally, the probability of a string X being emitted by an HMM M is computed as a sum over all possible paths by:

$$P(X|M) = \sum_{q_0 \ldots q_l} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k)\, P(q_k \uparrow x_k)$$

where q_0 and q_{l+1} are restricted to be q_1 and q_F respectively, and x_{l+1} is an end-of-string token [10].

While extracting the information the state sequence with the highest probability will be chosen [10]. That is;

$$V(X|M) = \underset{q_0 \ldots q_l}{\operatorname{argmax}} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k)\, P(q_k \uparrow x_k)$$

Learning to Perform Protein Name Extraction

To build an HMM for information extraction, first how many states the model should contain and what transitions between states should be allowed must be decided. We can learn the model structure from training data. Training data with tagged protein names can be used to build the model. After producing the model, we can apply it to the test data and chose the state sequences with the highest probability as protein names.

Training is a matter of scanning the training corpus and building the various probability tables needed for the protein name extraction task. Pseudocode for the training algorithm is shown in Figure-2.

```
while not eof (training_data) {
  Read training data from the training data set;
  Extract articles;
  Create a list of training words;
  Collect protein names;
}
for each word w in the training set {
  calculate frequencies for w;
}
for each word w_k in the training set {
  for each word w_k-1 in the training set {
    calculate the probability P(w_k | w_k-1)
    calculate the probability P(w_k | w_k-1)
  }
}
```

Figure 2: Training algorithm

In the learning algorithm, we first read the training data. Afterwards, we process the training data, collect information about the articles and extract the all words in the training data. At the same time, we process and extract all the tagged protein names. We produce statistical data about the words present in the training data: total number of occurrences, number of occurrences in a protein name, number of occurrence in a protein name as a first word, number of occurrence in a protein name as a last word and number of occurrence in a protein name as a single word. We use two linked-lists to keep word-list and protein name list. While inserting the words into the list, we determine the type of the word.

After collecting the statistical data about the words, we begin to produce four probabilities for each word: probability of being a first word in a protein name, probability of being a last word in a protein name, probability of being a single protein name and probability of being a word in a protein name.

Afterwards, we scan all words and for each word we calculate: a) *the count of coming a word w_k after another word w_{k-1}* b) *the count of coming a word w_k after another word w_{k-1} in a protein name*. At the same time, for each word-type we calculate: a) *the count of coming a word-type wt_k after another word-type wt_{k-1}* b) *the count of coming a word-type wt_k after another word-type wt_{k-1} in a protein name*. We use two matrices for these calculations. At this point, the training process is completed.

After building necessary probability tables, test algorithm can estimate the protein names in the test corpus. During testing, an estimate is produced for every fragment in the test data.

In the test algorithm, we first read the test data. After reading, we process the test data, collect information about the articles and extract the all words in the test data. Afterwards, for each fragment in the test data we calculate the likelihood probabilities. We use sliding-window technique to determine fragments. Each word in the window represents a state in the HMM. While calculating these probabilities, we use the word-type probabilities. Thus, we achieve the generalization task.

Putting the probabilities generated during the training into the below formula

$$V(X|M) = \underset{q_0 \ldots q_l}{\operatorname{argmax}} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k)\, P(q_k \uparrow x_k)$$

we can make estimates for each fragment. Fragments with the maximum likelihood are extracted as protein names. Pseudocode for the training algorithm is shown in Figure-3.

```
while not eof (test_data) {

  Read training data from the training data set;

  Extract articles;

  Create a list of test words;

}

prob = maxProb = 0;

for k=1 to NO_OF_TEST_WORDS{

  for l=k+1 to k+WIN_SIZE {

    prob= P(w_{k+1} | w_k) P(w_{k+2} | w_{k+1}).....P(w_l | w_{l-1})

    if prob>maxProb then maxProb=prob;
  }

  if maxProb>threshold
        then extract the fragment;
}
```

Figure 3: Test algorithm

EXPERIMENTAL EVALUATION

Methodology

In this section, we present how our model performs the extraction task in terms of precision and recall. We begin this section by explaining the methodology followed in our experiments.

To evaluate the effectiveness of our model, we conducted a series of comparative experiments. We used the same annotated corpora used in [7]. The details of the corpora were presented. In our experiments, two protein names are considered a match if they consist of the same character sequence in the same position in the text.

We measured precision (percentage of extracted names that are correct), recall (percentage of correct names that are found), and F-measure (harmonic mean of precision and recall); as is commonly done in the MUC evaluations. The metrics and their calculation methods are shown below:

$$\text{Precision} = \frac{\text{\# of correct protein names}}{\text{\# of extracted protein names}}$$

$$\text{Recall} = \frac{\text{\# of correct protein names}}{\text{\# of protein names to extract}}$$

$$\text{F-measure} = \frac{(\beta^2 + 1.0)\, PR}{(\beta^2 P) + R}$$

The parameter β determines how much to favor recall over precision. We set β parameter to 1 (β =1) to make precision and recall equally-weighted.

Results

Figure-4 shows the precision-recall graph for the experiment. In the graph, we show the curve indicating the precision for each achievable level of recall.

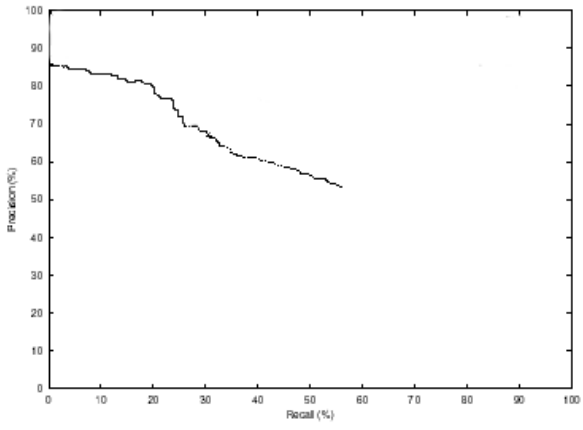

Figure 4: Precision - Recall graph

Maximum F-measure value measured during the experiment and precision-recall values for that point is shown in Table-2.

Precision	Recall	F-measure
0.5473	0.5448	0.5460

Table 2: Maximum F-measure

Experiments show that the use of word-type generalization increases the performance of HMM.

We tried to improve the performance and made some changes in the test algorithm. In the test algorithm, instead of using word-type probabilities in every case –whether or not the word exists in the training set– we try to use the maximum of the word-type probability and word probability.

Figure-5 shows the precision-recall graph for the derived method. In the graph, we show the curve indicating the precision for each achievable level of recall. Maximum F-measure value measured during the experiment and precision-recall values for that point is shown in Table-3.

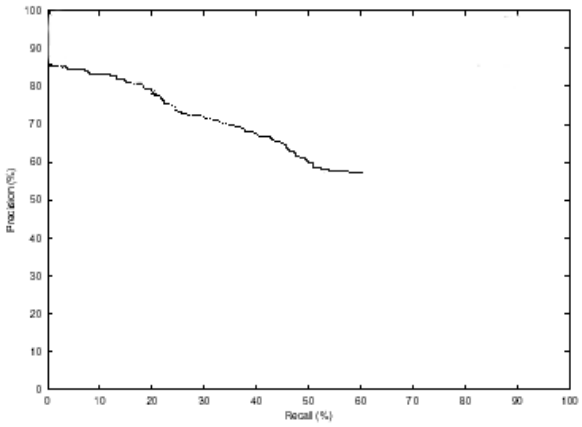

Figure 5: Precision - Recall graph for the derived method

Precision	Recall	F-measure
0.5575	0.6037	0.5797

Table 3: Maximum F-measure for the derived method

We had a little improvement by changing the algorithm. Both precision and recall values show the improvement.

CONCLUSION

This study has showed the initial results on the extracting protein names from Medline abstracts using HMMs. One of the most important conclusions from this project is the high importance of model design and generalization method in HMMs applied to protein name extraction.

REFERENCES

[1] Bunescu, R., Ge, R., Kate, R. J., Mooney, R. J., Wong, Y. W., Marcotte, E. M. and Ramani, A. K. Learning to Extract Proteins and their Interactions from MedlineAbstracts. In Proceedings of the ICML-2003 Workshop on Machine Learning in Bioinformatics, pp. 46–53, August 2003.

[2] Bunescu, R., Ge, R., Kate, Marcotte, E. M., Mooney, R. J., Ramani, A. K. and Wong, Y. W. Comparative experiments on learning information extractors for proteins and their interactions. Special Issue in the Journal Artificial Intelligence in Medicine on Summarization and Information Extraction from Medical Documents, 31, 2004.

[3] Cardie, C. (1997). Empirical methods in information extraction. AI Magazine, 18, pp.65-79.

[4] Collier, N., Park, H. S., Ogata, N., Tateishi, Y., Nobata, C., Ohta, T., Sekimizu, T., Imai, H., Ibushi, K. and Tsujii, J. The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers. In Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pp.271_272, June 1999.

[5] Fukuda, K., Tsunoda. T., Tamura, A. and Takagi. T. (1998) Toward information extraction: identifying protein names from biological papers. In Proceedings of the Pacific Symposium on Biocomputing (PSB98), pp. 705-716.

[6] Humphreys K., Demetriou G., and Gaizauskas, R. (2000) Two applications of information extraction to biological science journal articles: enzyme interactions and protein structures. In Proceedings of the Pacific Symposium on Biocomputing (PSB2000) , pp. 502-513.

[7] Olsson, F., Eriksson, G., Franzén, K., Asker, L. and Lidén, P. “Notions of Correctness when Evaluating Protein Name Taggers”. In Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002), Taipei, Taiwan, 24 August - 1 September.

[8] Ray, S., Craven, M. (2001). Representing sentence structure in hidden Markov models for information extraction. Proc. of 17th Intl. Joint Conf. on Artificial Intelligence (IJCAI-2001) (pp. 1273-1279). Seattle, WA.

[9] Raychaudhuri, S., Chang, J. T., Sutphin, P. D., and Altman, R. B. (2002). Associating genes with gene ontology codes using a maximum entropy analysis of biomedical literature. Genome Research, 12, pp. 203-214.

[10] Seymore, K., McCallum, A., and Rosenfeld, R. Learning hidden Markov model structure for information extraction. In Working Notes of the AAAI Workshop on Machine Learning for Information Extraction, pp. 37–42. AAAI Press, 1999.

[11] Tanabe, L., and Wilbur, W. J. (2002). Tagging gene and protein names in full text articles. Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain. pp. 9-13.

[12] Tanabe, L., and Wilbur, W. J. (2002). Tagging gene and protein names in biomedical text. Bioinformatics, 18, pp. 1124 -1132.