Special Issue in the Journal Artificial Intelligence in Medicine on Summarization and Information Extraction from Medical Documents

Comparative Experiments on Learning Information Extractors for Proteins and their Interactions

Razvan Bunescu [a,1] Ruifang Ge [a,1] Rohit J. Kate [a]
Edward M. Marcotte [b,1,2] Raymond J. Mooney [a,*,1,3]
Arun K. Ramani [b] Yuk Wah Wong [a,4]

[a] *Department of Computer Sciences, University of Texas, Austin, TX 78712, USA*
[b] *Institute for Cellular and Molecular Biology and Center for Computational Biology and Bioinformatics, University of Texas, Austin, TX 78712, USA*

Abstract

Automatically extracting information from biomedical text holds the promise of easily consolidating large amounts of biological knowledge in computer-accessible form. This strategy is particularly attractive for extracting data relevant to genes of the human genome from the 11 million abstracts in Medline. However, extraction efforts have been frustrated by the lack of conventions for describing human genes and proteins. We have developed and evaluated a variety of learned information extraction systems for identifying human protein names in Medline abstracts and subsequently extracting information on interactions between the proteins. We demonstrate that machine learning approaches using support vector machines and maximum entropy are able to identify human proteins with higher accuracy than several previous approaches. We also demonstrate that various rule induction methods are able to identify protein interactions with higher precision than manually-developed rules.

Key words: information extraction, text mining, machine learning, protein interactions, Medline

1 Introduction

An incredible wealth of biological information generated using biochemical and genetic approaches is stored in published articles in scientific journals. Summaries of more than 11 million such articles are available in the Medline database. However, retrieving and processing this information is very difficult due to the lack of formal structure in the natural-language narrative in these documents. Automatically extracting information from biomedical text holds the promise of easily consolidating large amounts of biological knowledge in computer-accessible form. Information extraction (IE) systems could potentially gather information on global gene relationships, gene functions, protein interactions, gene-disease relationships, and other important information on biological processes.

A number of recent projects [22,25,3,41,48,51,24,43,38,21] have focused on the manual development of IE systems for extracting information from biomedical literature. Unfortunately, manual engineering of information extraction (IE) systems for particular applications is a tedious and time-consuming process [14]. Each new type of information to be extracted requires a significant new engineering effort to develop specific extraction patterns for identifying this information. Human-developed rules are also rarely able to accurately capture all of the variety of formats and contexts in which the desired information can appear in natural-language documents.

Consequently, significant recent research in information extraction has focused on using machine learning techniques to help automate the development of IE systems [8,6]. A number of machine learning methods, including grammar induction, hidden Markov models, inductive logic programming, naive Bayes text categorization, and decision tree induction, have been used to help automate the development of IE systems. First, learning systems are trained on a corpus of documents in which human experts have tagged the desired infor-

* Corresponding Author. Tel: +1-512-471-9558; fax: +1-512-471-8885.
Email addresses: razvan@cs.utexas.edu (Razvan Bunescu), grf@cs.utexas.edu (Ruifang Ge), rjkate@cs.utexas.edu (Rohit J. Kate), marcotte@icmb.utexas.edu (Edward M. Marcotte), mooney@cs.utexas.edu (Raymond J. Mooney), arun@icmb.utexas.edu (Arun K. Ramani), ywwong@cs.utexas.edu (Yuk Wah Wong).
URLs: http://www.cm.utexas.edu/faculty/Marcotte.html (Edward M. Marcotte), http://www.cs.utexas.edu/users/mooney (Raymond J. Mooney).
[1] Supported by grant IIS-0325116 from the National Science Foundation.
[2] Supported by the Welch Foundation (F-1515), the National Science Foundation (ITR-0219061), and the Texas Advanced Research Program.
[3] Supported by grant IIS-0117308 from the National Science Foundation.
[4] Supported by an MCD Fellowship from the University of Texas at Austin.

mation. Next, the IE systems induced from this supervised data are used to extract new information from novel test documents. Some projects on extracting information from biomedical literature have also employed such learning techniques [42,15,11,46,18,32,47,40].

We are exploring the use of a variety of machine learning methods to automatically develop IE systems for extracting information on gene/protein name, function and interactions from Medline abstracts. For our purposes, genes and proteins are interchangeable since, typically, there is a direct correspondence between proteins and the genes that code for them. We focus specifically on extracting information about human genes and proteins. Approximately 40,000 human genes are known from the sequences of the human genome [55,31], yet fewer than 5,000 are well characterized and likely to be described in the literature. Unlike other organisms, such as yeast or *E. coli*, human gene names have no standardized naming convention, and thus represent one of the most difficult set of gene/protein names to extract. For example, human genes/proteins may be named with standard English words, such as "light", "map", "complement", and "Sonic hedgehog". Names may be alphanumeric, may include Greek or Roman letters, may be case sensitive, and may be composed of multiple words. Names are frequently substrings of each other, such as "epidermal growth factor" and "epidermal growth factor receptor", which refer to two distinct proteins. It is therefore necessary that an information extraction algorithm be specifically trained to extract gene and protein names accurately.

In this paper, we present results on learning to extract human protein names and their interactions. We employ a variety of learning methods including pattern-matching rule induction (RAPIER) [7], boosted wrapper induction (BWI) [19], memory-based learning (MBL) [17], transformation-based learning (TBL) [5], support vector machines (SVMs) [53], and maximum entropy (MaxEnt) [1]. We present cross-validated results on identifying human proteins and their interactions by training and testing on a set of approximately 1,000 manually-annotated Medline abstracts that discuss human genes/proteins. Previous projects on extraction from Medline typically present results for a single method on somewhat smaller corpora with limited or no comparison to other methods. By contrast, we present uniform results of a wide variety of methods on a single, reasonably large, human-annotated corpus, thereby giving a broader picture of the relative strengths of different approaches.

2 Biomedical Corpora

2.1 Tagging of Medline Abstracts

In order to generate a corpus of training and test data for extracting protein names and protein interactions, we manually tagged approximately 1,000 abstracts (including the titles) from among the 11 million abstracts available in Medline. Tagging was performed using an existing IE-tagging tool [5] modified to enhance file handling and to retain negative examples. This program accepts a directory of files to be tagged and allows the user to tag them using a graphical interface based on a file of possible labels and writes the SGML tagged files into an output directory. Three annotated data sets were generated:

(1) 750 abstracts containing the word "human" were extracted from the Medline database and tagged for gene/protein names. 61.3% of the abstracts discussed gene/protein names, for a total of 5,206 names. An example of a tagged abstract is shown in Figure 1.
(2) 200 abstracts previously known to contain protein interactions were obtained from the Database of Interacting Proteins (DIP [56]) and tagged for both 1,101 protein interactions and 4,141 protein names. An example is shown in Figure 1.
(3) As negative examples for protein interactions were rare in (2), a set of 30 abstracts were manually selected such that they had sentences with more than one gene but the abstracts did not talk about any gene interactions.

We used data set (1) for testing protein names, and data sets (2) and (3) for testing protein interactions.

2.2 Rules used for Tagging

Due to the ambiguities involved in human gene/protein names and interactions it was necessary to develop a set of conventions for their consistent tagging. In the following discussion we indicate protein names by underlined text and their same subscript numbers indicate interaction between the proteins. Manual examination of many abstracts revealed several ambiguities, such as whether the organism names should be tagged (e.g. human delta catenin or human delta catenin), whether punctuation should be tagged (e.g. (LIGHT) or (LIGHT)), and whether generic protein family names should be tagged (e.g.

[5] URL: http://www-2.cs.cmu.edu/~kseymore/general_tagger.pl

PMID -- 9367879
TI -- A c - Cbl yeast two hybrid screen reveals interactions with 14 - 3 - 3 isoforms and cytoskeletal components .
PG -- 46 - 50 AB - The protein product of c - cbl$_{1,2,3}$ proto - oncogene is known to interact with several proteins , including Grb2$_1$, Crk$_2$ and PI3 kinase$_3$, and is thought to regulate signalling by many cell surface receptors .
The precise function of c - Cbl in these pathways is not clear , although a genetic analysis in Caenorhabditis elegans suggests that c - Cbl$_4$ is a negative regulator of the epidermal growth factor receptor$_4$. Here we describe a yeast two hybrid screen performed with c - Cbl in an attempt to further elucidate its role in signal transduction . The screen identified interactions involving c - Cbl$_{5,6}$ and two 14 - 3 - 3 isoforms , cytokeratin 18 $_5$, human unconventional myosin IC , and a recently identified SH3 domain containing protein , SH3 P17$_6$.
We have used the yeast two hybrid assay to localise regions of c - Cbl required for its interaction with each of the proteins . Interaction with 14 - 3 - 3 is demonstrated in mammalian cell extracts .
AD -- Trescowthick Research Laboratories , Peter MacCallum Cancer Institute .

Fig. 1. Abstract with all the proteins and interactions tagged. The protein names have been underlined and their same subscript numbers indicate interaction between the proteins.

armadillo protein p0071 or armadillo protein p0071). Such cases led to the following set of tagging conventions:

(1) As few extra characters as possible are tagged. Punctuation marks and plural characters are not tagged.
(2) Gene/protein names are tagged regardless of context, even when gene names are substrings of other gene names. (e.g. GITR ligand)
(3) Generic protein/gene families are not tagged, only specific names which could ultimately be traced back to specific genes in the human genome. (e.g. "Tumor necrosis factor" would not be tagged, while "tumor necrosis factor alpha" would be.)
(4) Tags for interacting proteins follow the same conventions as for other proteins. All stated instances of protein interactions are tagged, even when tags are nested. (e.g. human GITR$_1$ ligand (hGITRL$_1$))

3 Protein Name Identification

Named entity recognition (NER), identifying names of people, organizations, and places in text, is a well studied problem in information extraction from news articles. In recent years, machine learning approaches have become the standard in developing robust, accurate NER systems [2,49]. Biomedical applications have special types of named entities that are different from those typically addressed by existing NER systems. These include names of diseases, genes, proteins, organisms, organs, organelles, and other biological entities. In this section we explore the problem of recognizing references to human proteins using the tagged data described in the previous section.

3.1 IE Methods

3.1.1 Dictionary-based Extraction

The success of a protein tagger depends on how well it captures the regularities of protein naming as well as name variations. In the dictionary-based approach, we started with an extensive set of protein names extracted from two fairly comprehensive sources:

(1) The file `human.seq`, downloaded from the Human Proteome Initiative (HPI) of EXPASY. [6]
(2) The file `feb2002-tables.tar.gz`, downloaded from the Gene Ontology Database. [7]

Altogether, these dictionaries contain 42,172 gene/protein names (synonyms included). This collection of protein names, henceforth referred to as the original dictionary (OD), was further extended using a generalization procedure to obtain a generalized dictionary (GD). The aim was to extend the coverage of the original set, while at the same time trying to minimize any decrease in accuracy.

Generalizing a dictionary entry involved identifying those parts susceptible to change in new protein names, and replacing them with generic placeholders. Thus, we isolate and replace numbers with $\langle n \rangle$, Roman letters with $\langle r \rangle$ and Greek letters with $\langle g \rangle$. Figure 2 shows some examples of name generalizations.

In the GD-based extraction, we tag a textual n-gram as a protein name only if it is an instance of one of the generalizations from the generic dictionary. To

[6] URL: `http://us.expasy.org/sprot/hpi/hpi_ftp.html`
[7] URL: `http://www.godatabase.org/dev/database/archive`

Protein Name (OD)	*Generalized Name (GD)*	*Canonical Form (CD)*
interleukin-1 beta	interleukin $\langle n \rangle$ $\langle g \rangle$	interleukin
interferon alpha-D	interferon $\langle g \rangle$ $\langle r \rangle$	interferon
NF-IL6-beta	NF IL $\langle n \rangle$ $\langle g \rangle$	NF IL
TR2	TR $\langle n \rangle$	TR
NF-kappa B	NF $\langle g \rangle$ $\langle r \rangle$	NF

Fig. 2. Dictionary generalizations.

extend the coverage even more, we have created a canonical dictionary (CD) consisting of canonical forms of protein names. A canonical form is obtained from a generic form by stripping it of all generic tags, as can be seen in the examples from Figure 2. From the resulting set we filter out common English words whose presence could lead to a decrease in accuracy. Consequently, in the CD-based extraction, a textual n-gram is deemed as being a protein name if its canonical form is part of the canonical dictionary. Both GD and CD introduce spurious entries in the dictionary, leading to a decrease in precision. For instance, because "HT3" is an entry in OD, its generalization $HT\langle n \rangle$ will cover "HT 29", which is a cell line. Also, CD will match words that are classes of proteins, and not particular proteins, as is the case with "oncogene" which was derived as the canonical form of "oncogene 24P3". Because the tagging based on both the original and generic dictionary gave better results than other combinations (as shown in the first entry of Table 1), we used this particular dictionary-based tagger for supplying a pre-tagged input to some of the learning methods that will be discussed in the following sections.

3.1.2 Rapier

Rapier [7] is a rule learning algorithm that acquires unbounded patterns for extracting information from text. Each extraction rule consists of three parts: (1) a pre-filler pattern that matches text immediately preceding a filler (e.g. a protein name), (2) a filler pattern that matches the extracted substring, and (3) a post-filler pattern that matches the text immediately following the filler. Rapier begins with a most-specific set of rules and compresses the rule base by repeatedly replacing rules with more general ones.

To construct the initial rule base, most-specific patterns are created for each training example, specifying words for the filler, all words in the text preceding the filler, and all words in the text following the filler. To generate new rules, pairs of existing rules are randomly selected and their least-general generalizations created. Rapier starts with rules containing only generalizations of the filler patterns, and uses beam search to efficiently specialize the rules by adding pieces of the generalizations of the pre- and post-filler patterns of the seed rules, until the best rule in terms of information gain produces no spurious fillers when matched against the training examples. The best generalized

```
PMID -- 11529898
Lol p 1 is one of the most important allergens in grass pollen
extracts...
Lol p 1 is one of the most important allergens in grass pollen
extracts...
BBBB Lol EEEE BBBB p 1 EEEE is one of the most important
allergens in grass pollen extracts...
```

Fig. 3. Incorporating information from the dictionary-based tagger. The first sentence contains the correct tagging. The second sentence is the output of the dictionary-based tagger. The third sentence shows the input for RAPIER and BWI. The output tags of dictionary-based tagger have been transformed into special tokens BBBB and EEEE standing for begin and end of the tags respectively.

rule is then added to the rule base, and the process repeats until compression has failed more than a specified number of times.

To help RAPIER capture generalities that are not evident from the words alone, we supplied additional syntactic and semantic information to the learner in some of our experiments. First, we added part-of-speech (POS) tags to every word in the text. POS tags are potentially useful because certain types of words (e.g. cardinal numbers and proper nouns) are likely candidates of being parts of a protein name.

In another experiment, we included the output of the dictionary-based tagger (Section 3.1.1) in place of the POS tags in the form of special tokens (see Figure 3). By adding these tokens, we incorporated domain knowledge into the learning algorithm. At the same time, the learning algorithm can find general patterns that refine the output of the dictionary-based tagger.

3.1.3 Boosted Wrapper Induction

Boosted Wrapper Induction (BWI) [19] learns extraction rules composed only of simple contextual patterns called *wrappers* [29]. Although wrappers are highly accurate predictors of the start or end of a protein name, each of them has limited coverage since Medline abstracts do not exhibit a rigid structure. BWI circumvents this limitation by using boosting [20], which repeatedly learns simple, weak patterns that focus on the training examples for which the previous patterns have done poorly. The predictions of all learned patterns are then combined using a weighted voting scheme. The result is a boosted wrapper, which has been shown to be successful in several natural text domains.

To perform protein-name extraction using a boosted wrapper, every word boundary i in a Medline abstract is first given a *fore* score $F(i)$, which indicates

its likelihood of being the start of a protein name, and an *aft* score $A(i)$, which indicates its likelihood of being the end of a protein name. Then, the wrapper recognizes a text fragment (i, j) as a protein name if and only if $F(i)A(j)H(j-i) > \tau$, where $H(k)$ is a function that reflects the probability that a protein name has length k, and τ is a numeric threshold that controls the level of recall. By varying τ, we are able to perform extraction at different degrees of confidence.

In our experiments with BWI, we tested the usefulness of including the output of the dictionary-based tagger (Section 3.1.1) as part of the input of the learner, in the same way as it was done in Section 3.1.2.

3.1.4 Support Vector Machines

Support Vector Machines (SVMs) are one of the most recently developed classification methods [54]. They are well-founded in computational learning theory, and have been shown to generalize well in the presence of very many features. They are generally considered to be the currently best technique for text classification [26].

Assume that all m training examples consist of a vector of n features, and belong to either positive or negative class as follows: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, where $\mathbf{x}_i \in \mathbf{R}^n$ is the i-th feature vector and $y_i \in \{+1, -1\}$ is its class label. Then an SVM learns an optimal threshold function $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b, \mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R}$, which separates the training examples into two classes. An example $\mathbf{x}$ is classified as positive when $f(\mathbf{x}) > 0$, or negative when $f(\mathbf{x}) < 0$. A threshold function is optimal when the margin of separation between the two classes is maximal. It can be proven that the margin is maximized when the norm of $\mathbf{w}$ is minimized. This leads to a constrained quadratic optimization problem which can be exactly solved efficiently.

Since our tagged Medline abstracts do not contain any protein names that directly abut each other, we can reduce the NER problem to classification of individual words. First, an SVM classifier determines if each word is part of a protein name or not, by looking at the word itself and its surrounding context. Next, protein names are extracted by identifying the longest sequences of words that have been classified as parts of a protein name. Similar approaches have been applied successfully to the task of *text chunking*, which is identifying simple phrases such as non-recursive noun and verb phrases [45,49].

For each token, we built a feature vector consisting of the current word, the previous and the following N words. We also included POS tags generated by the Brill's tagger [8] and the output of the dictionary-based protein tagger

[8] URL: `http://www.cs.jhu.edu/~brill/RBT1_14.tar.Z`

(Section 3.1.1) for all $2N+1$ words. We ignored capitalization when preparing the feature vectors to avoid sparsity. To capture morphological similarities and alleviate the problem of unseen words, we included as features the last one, two, and three characters of each word in the feature vector, which we henceforth refer to as the *suffix features*. Inspired by the text chunking algorithm presented in [28], we included the class labels of the two preceding words as part of the feature vector. Since the class labels were not given in the test data, they were decided dynamically during the tagging of previous words. Because numerical values were needed, each word or tag in each position was a separate binary feature. For each extracted sequence of tokens, we used the minimal distance from the hyperplane $f(\mathbf{x}) = 0$ as a quantitative measure of confidence. For the inner product $\langle \mathbf{w}, \mathbf{x} \rangle$, we used $\mathbf{w}^T\mathbf{x}$, which resulted in a linear threshold function. It has been argued that most text categorization problems are linearly separable [26], so in our case a linear threshold function should suffice. We used version 5.0 of SVM^{light}, [9] which is highly efficient in dealing with sparse instances.

The training set for the token classification problem is highly imbalanced. Out of the 209,022 tokens in our corpus, only 10,175 of them (4.87%) are protein names. As pointed out by [27], the induced classifiers tend to be highly accurate on negative examples but also produce many false negatives which lead to low recall. By sampling the training set and feeding the learner with only negative examples surrounding the positive ones, we can shift the resulting hyperplane and potentially reduce the number of false negatives. Our experiments supported this claim and showed that we could attain very high recall at the expense of precision.

3.1.5 Maximum Entropy

Maximum Entropy [1] is a widely used method for inducing probabilistic classifiers. The classification problem is viewed in terms of a random process that produces an output value y from a finite set Y, based on a contextual information x, a member of a finite set X. In a tagging scenario, this means associating a tag y to each text token, whereas the context x can be derived from the text centered at the current token position. In maximum entropy modeling we are looking for a probability distribution $p(y|x)$ expressed in terms of a set of user specified features $f_i(x, y) \in F$:

$$p(y|x) = \frac{1}{Z(x)} exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

[9] URL: `http://svmlight.joachims.org/`

where $Z(x) = \sum_y exp\left(\sum_i \lambda_i f_i(x, y)\right)$ is a normalizing constant. Each feature f_i is a binary function based on the current context x and its proposed classification y.

In the case of maximum entropy tagging (henceforth referred to as MaxEnt), we distinguish among five types of tags in Y (as opposed to using only two tags, as was the case with SVMs):

- **S***(-tart)* - indicates the first token of a protein name
- **E***(-nd)* - indicates the last token of a protein name
- **C***(-ontinue)* - indicates a token strictly inside a protein name
- **U***(-nique)* - indicates the unique token of a protein name
- **O***(-ther)* - all other tokens (outside protein names)

We hypothesize that the task of tagging the first, the last, or the unique token of a protein name is slightly different from that of tagging other tokens inside a protein name, hence the extended set of tags

Name	*Feature Description*	*Feature Body*	
w	current word	$w(x_i) = \langle w \rangle$	& $y_i = \langle y \rangle$
pw	previous word	$pw(x_i) = \langle w \rangle$	& $y_i = \langle y \rangle$
nw	next word	$nw(x_i) = \langle w \rangle$	& $y_i = \langle y \rangle$
pos	POS, current word	$pos(x_i) = \langle pos \rangle$	& $y_i = \langle y \rangle$
$ppos$	POS, previous word	$ppos(x_i) = \langle pos \rangle$	& $y_i = \langle y \rangle$
$npos$	POS, next word	$npos(x_i) = \langle pos \rangle$	& $y_i = \langle y \rangle$
cf	word class (full)	$cf(x_i) = \langle cf \rangle$	& $y_i = \langle y \rangle$
cb	word class (brief)	$cb(x_i) = \langle cb \rangle$	& $y_i = \langle y \rangle$
$dict$	dictionary tag	$dict(x_i) = \langle dt \rangle$	& $y_i = \langle y \rangle$
pt	previous tag	$pt(x_i) = \langle y' \rangle$	& $y_i = \langle y \rangle$

Fig. 4. Feature Templates.

The abstracts are tokenized, segmented in sentences, and annotated with part-of-speech tags using the same tools as in Section 3.1.4. Then the model generates feature vectors by scanning each pair (x_i, y_i) in the training data using the feature templates given in Figure 4. We use a threshold of 3 as the minimum number of times that a feature should appear in the training data in order to be considered. The word class features *cf* and *cb* are based on the similar features introduced in [13]. Thus, for a character x we define $type(x)$ as 'A' if x is a upper-case letter, 'a' if x is a lower-case letter, '0' if x is a digit and x otherwise. The *cf* feature then is the current word with each character mapped to its type, while the brief version *bf* results from *cf* by removing repeating character types. For example, if "FGF1" is the current word, then *cf='AAA0'*, and *bf = 'A0'*. Another special feature is *pt*, based on the tag assigned to the previous token. The dependence of the current tagging deci-

sion on the previous tag, unknown during testing, forces us to consider all possible tags for the previous token when tagging unseen data. For a particular token sequence (tokenized sentence), this will result in a potentially very large set of possible taggings. The classical approach is to use a Viterbi-like algorithm for finding the most likely sequence of tags. To each of the resulting extractions we associate a confidence measure. The exact calculation of the two confidence measures $conf_{avg}$ and $conf_{min}$ is described in Figure 5. The Viterbi and forward procedures used therein are similar with those used for Hidden Markov Models [44]. Varying the confidence level will later allow us to trade off between precision and recall (see Section 3.2.2).

$conf(W, u, v)$	
Input:	W, a sequence of tokens $w_1, w_2, \ldots, w_T$ $[u, v]$, an extraction span with $1 \leq u \leq v \leq T$
Use the forward procedure on W with $p(y\|x)$ to compute: $\alpha_t(y) = p(y_t = y\|W)$, where $1 \leq t \leq T$ and $y \in \{S, C, E, U, O\}$ **if** $u = v$ $conf_{avg}(W, u, v) = conf_{min}(W, u, v) = \alpha_u(U)$ **else** $p_u = \alpha_u(S)$ $p_{u+1} = p(y_{u+1} = C\|W, y_u = S)$ $\ldots$ $p_v = p(y_v = E\|W, y_{v-1} = C)$ $conf_{avg}(W, u, v) = \frac{1}{v-u+1} \sum_{t=u}^{v} p_t$ $conf_{min}(W, u, v) = \min_{u \leq t \leq v}(p_t)$	

Fig. 5. Extraction Confidence.

We chose to take either the minimum or the average as we were targeting a length-independent measure. Also, when using the average function, or the minimum function, one has to ensure that the quantities involved have a similar interpretation and consequently can be safely combined. In our case, we can view the value $\alpha_t(y)$ as another transition probability, namely the probability of reaching state y at time step t from a special state encoding the beginning of the sentence.

One drawback of using the Viterbi algorithm is that by focusing on the most likely sequence of tags, the program is missing many low confidence extractions that might help in extending the recall endpoint. When applied on test data, the Viterbi algorithm, augmented with the confidence measure $conf_{min}$, results in a maximum recall of 47.76%. To further extend it, we use the greedy algorithm from Figure 6 on all token sequences appearing between two consecutive Viterbi extractions, thus obtaining additional extractions compatible with the set of proteins already extracted through the Viterbi procedure (two extractions are compatible if they do not overlap). All the results that presuppose using a confidence measure are based on $conf_{min}$, which does a better

job at extending the recall endpoint.

$greedy\_extract(W, u, v)$	
Input:	W, a sequence of tokens $w_1, w_2, \ldots, w_T$ $[u, v]$, an extraction domain with $1 \leq u \leq v \leq T$

if $u > v$
  return $\emptyset$
else
  $[l, r] = argmax_{[l,r] \subseteq [u,v]} conf(W, l, r)$
  $LE = greedy\_extract(W, u, l-1)$
  $RE = greedy\_extract(W, r+1, v)$
  return $LE \cup \{[l, r]\} \cup RE$

Fig. 6. Greedy Extraction.

We base our Maximum Entropy approach on the opennlp.maxent package, [10] version 2.1.0, which uses the Generalized Iterative Scaling algorithm [16] for estimating the parameters of the log-linear model.

3.1.6 Existing Protein Name Identification Systems

We also tested two existing protein name identification systems. The first one is KEX version 1.21, which is based on the PROPER algorithm described in [22]. It consists of a set of hand-built pattern matching rules which makes use of part-of-speech information given by the Brill's tagger. Without depending on any protein-name dictionaries, KEX has been reported to achieve 94.70% precision and 98.84% recall on a corpus of 80 abstracts on SH3 and signal transduction domains.

The second system is ABGENE, introduced in [50]. ABGENE uses a transformation-based tagger to produce an initial tagging. Then it employs a number of dictionaries and contextual rules to weed out false positive and recover false negative. It was tested on a corpus consisting of the complete set of abstracts introduced into Medline between June 15 and September 24, 2001, and was reported to give good results.

3.2 Experimental Results

We begin this section by explaining the methodology followed in our experiments. We present next the quantitative results of the IE methods used for extracting protein names. The section ends with a comparative analysis of the results.

[10] URL: `http://maxent.sourceforge.net/`

3.2.1 Experimental Methodology

The 750 Medline abstracts annotated with protein tags were tokenized using simple pattern rules developed for the Penn Treebank project [35]. For programs requiring sentence-segmented input, we used the sentence segmenter from the KEX tagger with additional rules for bulleted lists. For those learning algorithms requiring POS tags, we used Brill's POS tagger, which we trained by using 10,000 untagged Medline abstracts as the training set. Those abstracts were obtained the same way we did for the 750 abstracts. No stemming or stopword filtering was performed during the experiments. Capitalization was retained unless otherwise specified.

We performed ten-fold cross validation on each learning algorithm with a particular parameter setting. This provides average performance over ten random trials, each training on 90% of the data and testing on the remaining 10%. Each extracted protein name in the test data was compared to the human-tagged data, with the positions taken into account. Since ABGENE provides no positional information, we assume that all occurrences of its extracted strings are recognized as protein names. Two protein names are considered a match if they consist of the same character sequence in the same position in the text. This detects circumstances where common English words are incorrectly recognized as protein names (e.g. "light", "at"), and ensures that all references to each protein are recognized. 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) [14].

3.2.2 Quantitative Results

Table 1 summarizes results for the protein taggers presented in Section 3.1, along with any additional sources of information used. We also include results obtained with two additional taggers: one using transformation-based learning (TBL) [5], and another based on the k-nearest neighbor (k-NN) method in which classification is done by extrapolation from the k most similar training examples. For systems that output confidences that allow trading-off precision and recall (i.e. BWI, k-NN, SVM and MaxEnt), results are presented for the maximum achievable recall or the best F-measure.

For ease of comparison, we show recall-precision curves in Figure 7, using the version of each system that gave the best F-measure (as shown in bold in Figure 1). For those IE methods that output extraction confidences, we show curves indicating the precision for each achievable level of recall. Single recall-precision points are shown for all other methods.

Given that the MaxEnt approach achieves the best results on the 750 abstracts dataset, we applied it on the 230 abstracts from the interactions dataset,

IE Methods and Additional Information Used	*Precision*	*Recall*	*F-measure*
Dictionary-based			
original dictionary	56.70%	27.24%	36.80%
plus generalized dictionary	62.27%	45.85%	**52.81%**
plus canonical dictionary	41.88%	54.42%	47.33%
RAPIER			
words only	76.11%	9.97%	17.63%
part-of-speech	70.84%	11.05%	19.12%
dictionary-based tagger	74.49%	12.22%	**21.00%**
BWI (300 iterations, 2 lookaheads, max. recall)			
words only	70.67%	11.52%	19.81%
dictionary-based tagger	71.01%	24.06%	**35.94%**
k-NN ($k = 1, N = 2$)			
part-of-speech	34.66%	40.66%	37.42%
dictionary-based tagger	47.30%	47.82%	**47.56%**
TBL			
words only	47.08%	36.65%	41.22%
dictionary-based tagger	56.80%	34.62%	**43.02%**
SVM ($N = 2$, full training set, max. recall)			
preceding class labels	69.16%	19.74%	30.72%
preceding class labels and part-of-speech	70.18%	19.72%	30.79%
preceding class labels and dictionary-based tagger	65.00%	45.43%	53.48%
with additional suffix features	70.38%	44.49%	**54.42%**
MaxEnt ($N = 1$, Viterbi w/o greedy extraction, max. recall)			
w/o dictionary	71.10%	42.31%	53.05%
with dictionary	73.37%	47.76%	**57.86%**
with dictionary, two tags only (I,O)	66.41%	44.74%	53.46%
KEX	14.68%	31.83%	**20.09%**
ABGENE	32.39%	45.87%	**37.97%**

Table 1
Performance of protein taggers in various settings.

our aim being to feed these automatically tagged abstracts to the interaction extraction program (see Section 4.2.2 for overall results of the combined approach). The tagging performance on the interaction dataset is shown in Figure 8 (a).

3.2.3 Discussion of Results

Overall, the results show limited utility of POS tags. The use of POS tags in RAPIER, k-NN, and SVM does not improve F-measure significantly according to a paired t-test ($p > 0.05$). While the dictionary-based tagger barely im-

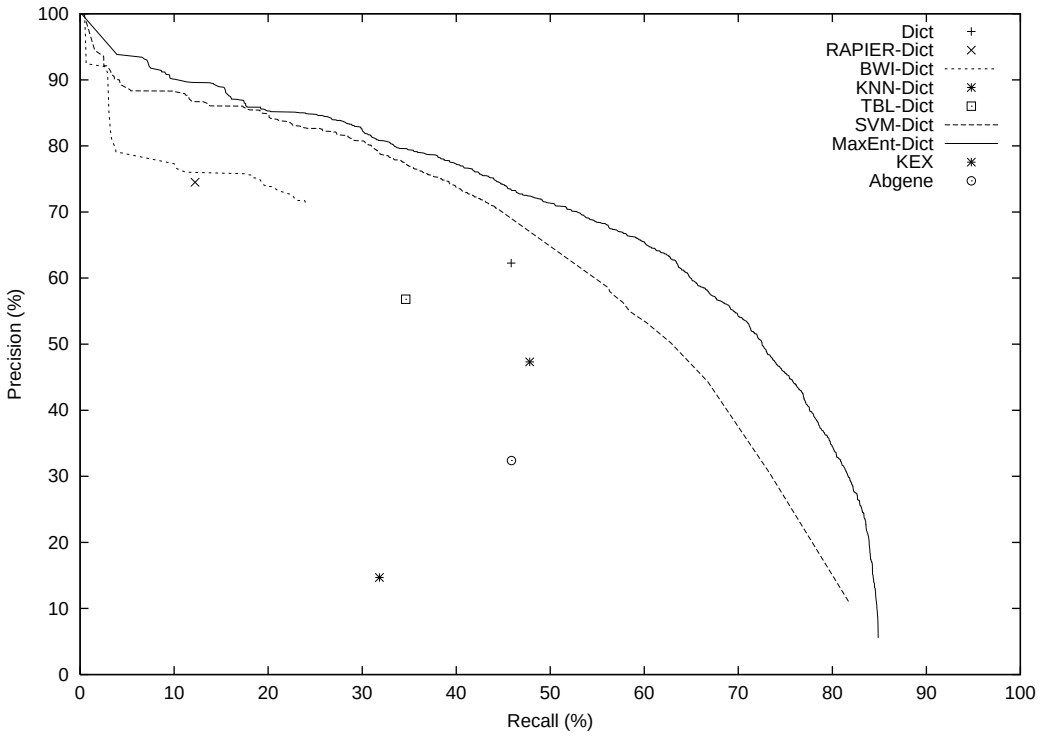
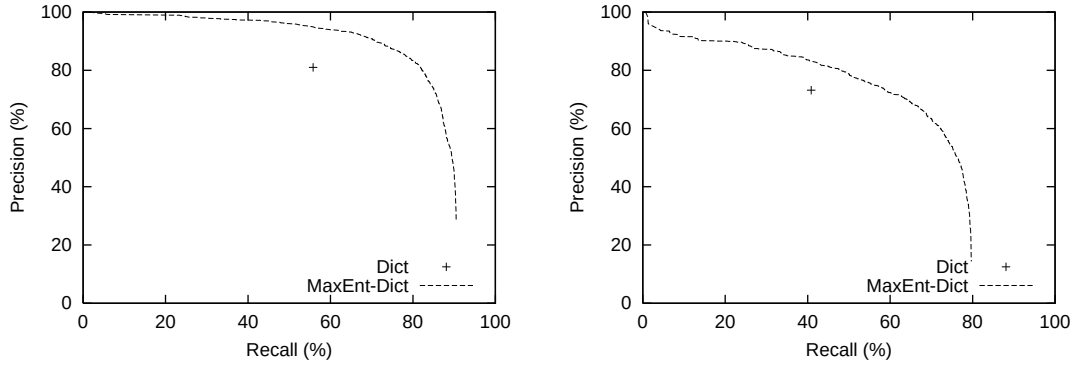


Fig. 7. Precision-recall curves for protein taggers on the 750 abstracts dataset.

proves F-measure for RAPIER and TBL, it is useful for the rest of the learning methods to different extents. It improves both precision and recall for BWI, k-NN and MaxEnt, while for SVM it hurts the precision slightly, but this is outweighed by a larger gain in recall.



(a) Interaction dataset (230 abstracts) (b) YAPEX dataset (200 abstracts)

Fig. 8. Performance of MaxEnt protein tagger on two other datasets.

Out of all the learning methods tested here, SVM and MaxEnt achieve a significant improvement over the dictionary-based tagger in terms of F-measure. Another advantage is that both are able to achieve arbitrarily high precision by adjusting the confidence level. This is possible because the extraction

confidence is highly correlated with the probability of correctness. Since high precision is needed to extract accurate knowledge from text, this is a significant contribution.

We have also included results for MaxEnt with a tagging scheme based on two tags only. The difference in performance validates our initial hypothesis suggesting the use of more than one tag for tokens inside a protein name.

All of our IE methods perform significantly better than two existing protein taggers, KEX and ABGENE. Given that these systems were developed for different distributions of proteins, this is not surprising; however it does illustrate the relative difficulty of identifying human proteins. The hand-built rules used in KEX were developed and tested on a rather confined set of proteins different from the human proteins in our data. ABGENE uses a version of TBL to learn a protein tagger; however, the specific tagger we obtained was not trained specifically for human proteins. Our own TBL system is more indicative of the performance of this approach when specifically trained for human proteins; however, note that many of the other learning approaches perform better than TBL.

As shown in Figure 8 (a), the performance on the interaction dataset is a lot better than on the protein dataset, and this is also reflected in the results of the dictionary-based tagger. There are two main reasons for this significant difference:

(1) The protein dataset has been manually tagged by 9 people, in just one pass. After analyzing the results, we have discovered significant tagging inconsistencies which clearly affected the learning performance. On the other hand, the interaction dataset has been tagged by one person only, resulting in a more consistent tagging.
(2) Each of the 230 interaction abstracts contains at least two proteins, due to the particular selection process described in Section 2.1. This results in a significant bias which is captured by the learning algorithm. Comparatively, 38.7% of the 750 abstracts dataset contain no proteins, while many of the same abstracts include various names for cell lines, or amino acids, names which are very similar with protein names, making the task of recognizing proteins more realistic, but at the same time harder.

We have also tried our protein name extraction systems on the Yapex [11] dataset (200 Medline abstracts) and the results (Figure 8 (b)) are comparable with those obtained on our corpus of 750 abstracts (Figure 7).

[11] URL: `http://www.sics.se/humle/projects/prothalt/`

SHPTP2$_{interactor}$ [interacts with another signaling protein ,]$_{interfiller}$ Grb7$_{interactee}$.

Fig. 9. Interactor, interactee and interfiller

4 Protein Interaction Extraction

Identifying *relations* between named entities stated in text is a more difficult IE problem that only recently has attracted significant attention in research on extraction from new articles. The current ACE (Automated Content Extraction) program at the National Institute of Standards and Technology (NIST) [37] is focused on identifying various social, action-role, part-of, and locational relations between named entities. Several projects have focused on extracting relations from biomedical text, such as identifying gene-disease relations, subcellular localizations, or protein interactions [15,46,18,25,3,41,48,51,34]. This section discusses our work on identifying human-protein interactions assuming that the proteins themselves have already been tagged, and shows that machine-learning systems out-perform human-written extraction rules with respect to providing a wider range of precision and recall.

4.1 IE Methods

4.1.1 Rapier *and Boosted Wrapper Induction*

In order to adapt slot-filling IE systems that extract individual entities (like Rapier and BWI) to the problem of extracting *relations*, we developed two approaches. The first approach we call the *Interfiller* approach. Given two tagged entities participating in a relationship, the text fragment between them is called the interfiller (see Figure 9). If a slot-filling IE system extracts an interfiller, the tagged entities before and after it can be extracted as participating in the targeted relation.

The second approach we call the *Role-filler* approach. In this approach, we extract the two related entities independently into different role-specific slots. For protein interactions, we named the roles *interactor* and *interactee* (see Figure 9). There might be many interactors and interactees extracted in one sentence, we then decide which of them participate in a relationship using the following heuristics, assuming that all interacting proteins appear in the same sentence. (1) The interactors and interactees appearing in the same sentence form a sequence of role fillers. This sequence is separated into segments at the points where an interactee is immediately followed by an interactor. Interactors and interactees can only be paired within the same segment. (2) Each interactor is associated with the next occurring interactee in the segment. (3)

[These | Here | have |,] (4) [data | we | previously | the | wild] (1) [suggest | show | reported | transcription |-] (2) [that | factor | type | of] (15) PROT (14) [surface | of | - | with | bound| activate] (0) PROT (27) .

Fig. 10. Sample protein-extraction rule learned by ELCS. Token PROT stands for protein name.

If there are fewer interactors (interactees) than interactees (interactors) in the segment, use the last interactor (interactee) in constructing the remaining pairs. In our human-tagged interaction corpus, assuming interactors and interactees are properly tagged, this approach identifies all the interacting pairs with 99.2% accuracy.

Both of these approaches have been used to train BWI (Section 3.1.3) to extract interacting proteins, and the Role-filler approach has been used to train RAPIER (Section 3.1.2) to extract interactions. RAPIER could not learn to extract interfillers successfully, since, in the worst case, the time complexity of its generalization algorithm can grow exponentially in the length of a filler. Since extracted entities are usually fairly short, this is typically not a problem in standard slot-filling IE. However, the long interfillers in many protein interactions prevented us from running RAPIER with the Interfiller approach.

4.1.2 Extraction using Longest Common Subsequences (ELCS)

We have also developed a new method for directly learning patterns for extracting relations between previously tagged entities. Blaschke *et al.* [3,4] manually developed rules for extracting interacting proteins. Each of their rules (or frames) is a sequence of words (or POS tags) and two protein-name tokens. Between every two adjacent words is a number indicating the maximum number of intervening words allowed when matching the rule to a sentence. Here we describe a new method ELCS (Extraction using Longest Common Subsequences) that automatically learns such rules.

ELCS' rule representation is similar to that in [3,4], except that it currently does not use POS tags, but allows disjunctions of words. Figure 10 shows an example of a rule learned by ELCS. Words in square brackets separated by '|' indicate disjunctive lexical constraints, i.e. one of the given words must match the sentence at that position. The numbers in parentheses between adjacent constraints indicate the maximum number of unconstrained words allowed between the two (called a *word gap*). A sentence matches the rule if and only if it satisfies the word constraints in the given order and respects the respective word gaps.

A sentence in the training data may contain more than two proteins and more

than one pair of interacting proteins. In order to extract the interacting pairs, the rules should be trained to pick out exactly the interacting proteins from the sentences. To do this we replicate the sentences having n proteins ($n > 2$) into C_2^n sentences such that each one has exactly two of the proteins tagged, with the rest of the protein tags omitted. If the tagged proteins interact, then the replicated sentence is added to the set of positive sentences, otherwise it is added to the set of negative sentences. During testing, a sentence having n proteins ($n > 2$) is again replicated into C_2^n sentences in a similar way. If such a replicated sentence matches one of the rules, then the system extracts the two proteins tagged in that sentence as interacting proteins.

ELCS induces rules using a bottom-up approach. Rule induction starts with maximally specific rules for each positive sentence which contain all the words in the sentence with zero-length word gaps. These are then repeatedly generalized to form more general rules until the rules become overly general and start matching negative sentences. We have developed three methods for generalizing rules. The first simple method to produce a generalization of two rules is to find the *longest common subsequence* (LCS) of words between them. Efficient algorithms for computing an LCS are presented in [10,23]. After finding the LCS between two rules, we determine the size of word gaps between every two adjacent words in their LCS as the larger of the number of words plus the sum of existing word gaps between the two LCS words where they are found in the original two rules.

Our second approach to generalization uses *edit distance* (ED) [23] and creates more specific rules that contain disjunctive constraints. The most common edit distance is Levenshtein distance [33], defined as the minimum number of edit operations (adding, deleting, or replacing an item) required to convert one sequence into another. We use the minimal edit-operation sequence obtained when computing Levenshtein distance to generalize two rules. We preserve the common word constraints between the rules, make disjunctions of constraints when one item is replaced by another in the edit sequence, and drop constraints that are added or deleted in the edit sequence. Finally, we introduce word gaps using the method described for the LCS-based generalization.

The third generalization method finds all common sequences between the two rules and considers their *conjunction* (CJ) as the generalization. Unlike the previous two methods, this method is associative, i.e. we get the same generalization of a set of rules irrespective of the order in which we generalize two of them at a time. If there is any common pattern among the base rules then this property guarantees that the pattern will also appear in the generalization (note that it is possible to lose such a common pattern while taking LCS of two rules at a time). Word gaps are then introduced as in the previous two methods. Figure 11 shows generalization of two sentences obtained by each of these methods.

Sentence 1: The self - association site appears to be formed by interactions between helices 1 and 2 of beta spectrin$_1$ repeat 17 of one dimer with helix 3 of alpha spectrin$_1$ repeat 1 of the other dimer to form two combined alpha - beta triple - helical segments .

Sentence 2: Title - Physical and functional interactions between the transcriptional inhibitors Id3$_2$ and ITF - 2b$_2$.

Generalization using longest common sequence (LCS):
- (7) interactions (0) between (5) PROT (9) PROT (17) .

Generalization using edit-distance (ED):
[self|Title] (0) - (4) [be|Physical] (0) [formed|and] (0) [by|functional] (0) interactions (0) between (2) [and|the] (0) [2|transcriptional] (0) [of|inhibitors] (0) PROT (8) [of|and] (0) PROT (17) .

Generalization using conjunctions (CJ):
{ - (7) interactions (0) between (5) PROT (9) PROT (17) . } $\bigwedge$ {- (11) and (6) PROT (9) PROT (17) . }

Fig. 11. Generalizations of two sentences using different methods. Protein names have been underlined and same sub-script numbers indicate interactions between them. Token 'PROT' stands for protein name.

Using one of these generalization methods, a greedy-covering, bottom-up rule-induction method is used to learn a small set of rules that cover all the positive sentences without covering many negative ones. We use an algorithm similar to beam search and consider only the r best rules for generalization at any time. We start with r randomly selected positive examples. These r rules are generalized with one of the remaining positive examples to obtain r more rules. Out of these $2r$ rules we select r rules with the highest confidence level and allow further generalization with the remaining positive examples. After iterating over the remaining positive examples in this way, the r best rules are finally included in the set of learned rules and the positive examples covered by them are removed. The entire process is repeated till we exhaust the set of positive examples.

We measure the confidence levels of our rules using m-estimate [9] which is a measure of expected accuracy of a rule. It is defined as: $\textit{confidence level}(\textit{rule}) = \frac{p+m.p^+}{p+n+m}$, where p and n are the number of positive and negative examples covered by the rule, p^+ is the prior probability of positive examples and m is a parameter which should be set according to the amount of noise in the data. We set p^+ as the fraction of examples in the training data which are positive and set m based on pilot studies.

The generalizations obtained using any of the methods may result in rules that

interactions (0) between (4) PROT (0) and (4) PROT (16) .

PROT (0) / (0) PROT (10) heterodimers (36) .

[binding | substitution | AB | addition | Interestingly | TI | interactions] (0) [of | - | ,] (3) PROT (19) [to | for | : | same | with] (10) PROT (30) [nM | binding | 1 | CDK6 | CCR8 | death] (9) .

[linker | TI | armadillo | b558 | of] (0) [- | , | a] (5) PROT (13) [and | / | with | to | containing] (0) PROT (2) .

{, (11) PROT (25) and (8) to (9) PROT (66) .} $\bigwedge$ {, (11) PROT (16) bind (18) PROT (66) .}

{, (10) PROT (5) for (7) PROT (9) .} $\bigwedge$ {, (10) PROT (4) binding (6) PROT (9) .}

Fig. 12. Some example rules learned by ELCS; the first two were learned using LCS generalization, the next two using ED generalization and the last two using CJ generalization.

do not contain two protein-name tokens. This is fine for extracting protein interactions because we always apply the rules to sentences containing exactly two protein names (if they contain more than two protein names then we replicate the sentence as described earlier). However, constraining learned rules to contain two protein names is a useful bias. Therefore, we divide each of the training sentences in three parts: the portion of the sentence before the first protein name, the portion between the two protein names, and the portion after the second protein name. When we generalize two rules, we generalize these three parts separately. This ensures the rule will always contain two protein-name tokens. Figure 12 shows some sample rules learned by ELCS.

4.2 Experimental Results

4.2.1 Experimental Methodology

Medline abstracts were pre-processed as described in Section 3.2.1. All our systems for extracting interactions require sentence segmentation since only the proteins within a sentence are considered when identifying interactions. This constraint is satisfied by all interactions in our corpus because while manually tagging the corpus with interactions we did not find a single instance where the two interacting proteins were in different sentences. We also compared our systems with Blaschke *et al.*'s manually-written rules [4]. Since these rules require POS tags, we used Brill's POS tagger. We also tested a version of the human-written rules in which the POS tags are replaced by typical words

indicating interactions such as *activation*, *phosphorylation* or *interaction* for nouns and *activates*, *binds* or *phosphorylates* for verbs, similar to the approach in [3]. These manually-written rules were developed to capture the common ways of expressing protein interactions in natural language, like "protein A binds/interacts ... (with) protein B" (see the SUISEKI system overview in [4]). Such natural language constructions are general enough and not sensitive to any particular dataset. Hence we applied these rules on our dataset to make a direct comparison with our systems.

We did two experiments to evaluate the performance of protein interaction extraction. In both experiments the machine learning systems were trained using the manually tagged abstracts (see Section 2.1) with proteins and their interactions. The two experiments differ in the way we tested the systems. In the first experiment we provide manually tagged protein names to our systems and extract interactions among these proteins. This way we get a measure of how the protein interaction extraction systems alone perform independent of the protein name extraction systems. In the second experiment we first find protein names in the abstracts using our best system for protein name extraction, MaxEnt (see Section 3.2.2), and then extract interactions among these proteins. This gives a true measure of how our systems can perform at extracting protein interactions from completely untagged abstracts. Blaschke *et al.*'s manually-written rules also require protein names, in this experiment we also test those rules by providing them with our extracted protein names. For this experiment we chose the point on the MaxEnt's precision-recall curve (Figure 8 (a)) which gives 70% precision and about 90% recall for protein name extraction.

As in Section 3.2.1, performance is evaluated using ten-fold cross validation and measuring recall and precision. We consider an extracted interaction from an abstract correct only if both its proteins have been human-tagged as interacting with each other somewhere in that abstract. As the task of interest is only to extract interacting protein-pairs, in our evaluation we don't consider matching exact positions and every occurrence of interacting protein-pairs within the abstract. For those IE methods which output extraction confidences, if we extract more than one occurrence of interaction between two proteins then we combine their extraction confidences using the standard Noisy-Or method [39].

4.2.2 Quantitative Results

Figure 13 shows recall-precision results for protein-interaction extraction when tested on abstracts that have been manually tagged for protein names and Figure 14 shows the results when tested on abstracts in which protein names were tagged using our best protein name extractor. We plotted a precision-

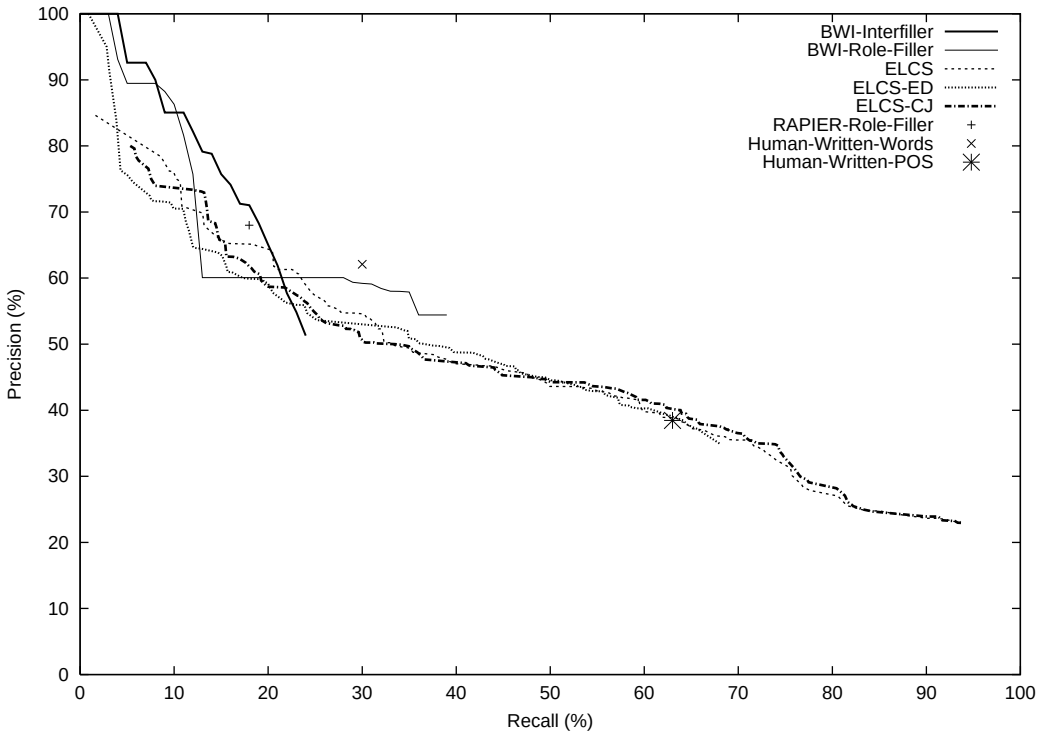


Fig. 13. Precision-recall graphs for protein interaction extraction using correct protein names.

recall curve for BWI by utilizing its extraction confidence, and for ELCS using the confidence levels of the rules which extract the interactions. Since RAPIER and human-written rules do not produce confidences, only a single recall-precision point is shown for each of them.

4.2.3 Discussion of Results

From Figure 13 it can be seen that BWI gives varying degrees of high precision, but its recall is generally quite low. RAPIER also gives relatively high precision but low recall. ELCS tends to give higher recall with only a modest decrease in precision compared to BWI and RAPIER. When we use the protein names extracted from our protein name extractor instead of the correct protein names, not surprisingly the performance of all the systems degrade but they still offer reasonable ranges of precisions and recalls (Figure 14).

These results demonstrate that machine learning systems can provide higher precisions than the human-written rules. In order to avoid over-loading human curators with too many false positives when extracting knowledge from large volumes of text, a general emphasis towards higher precision seems appropriate. The machine learning systems also offer a wide range of precision-recall trade-off which can be suitably utilized by a user depending on the needs of

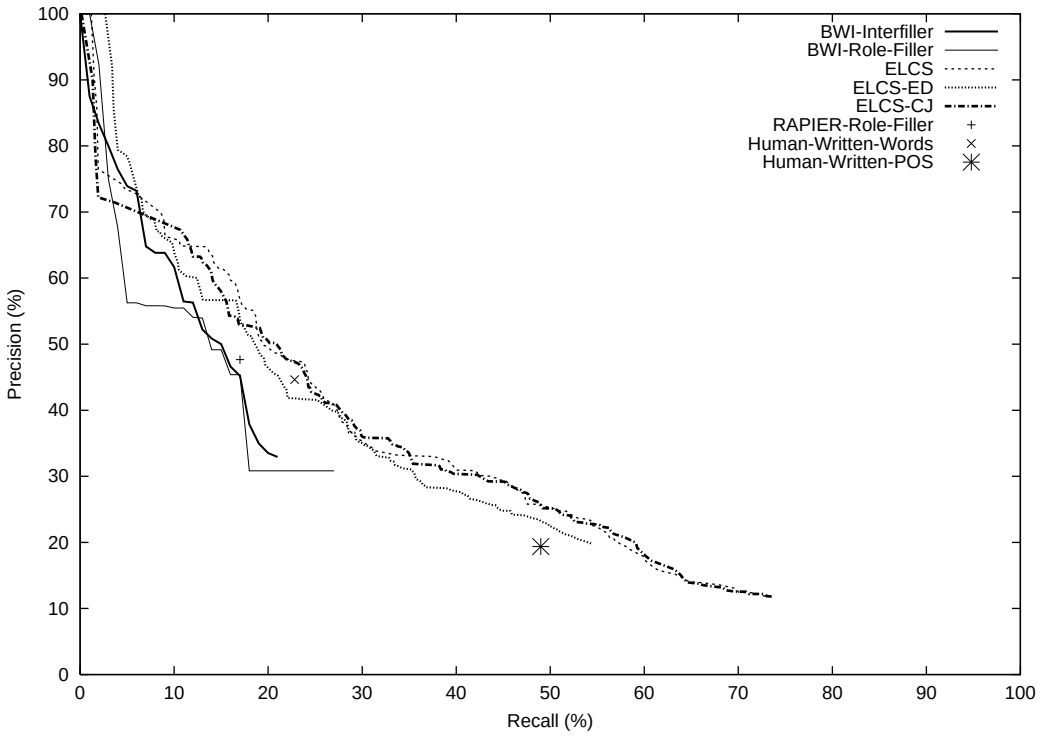


Fig. 14. Precision-recall graphs for protein interaction extraction using extracted protein names.

the application. The machine learning systems can also provide recalls higher than the best recall the human-written rules could provide. Hence the machine learning systems out-perform the manually-written rules in several ways.

5 Conclusions and Future Research

After comparing a number of methods for extracting human protein names and interactions, we obtained the best performance for protein tagging with a maximum entropy learning method that exploits a generalized protein-name dictionary. For extracting protein interactions, we found that several methods for learning extraction rules out-perform the hand-written rules in providing higher precisions and in offering a wider range along precision-recall trade-off. Token classification methods like k-NN, TBL, SVM, and MaxEnt are not directly applicable to extracting interactions; however, we plan to test HMMs on extracting interactions in the near future.

Clearly, the ability to extract human proteins and their interactions still needs significant improvement. We foresee improvement in three general areas: better training data, better learning methods, and better use of external knowledge.

Larger training sets are always beneficial to learning systems; however, manually tagging data is very time consuming. One alternative approach is to use existing knowledge to automatically produce *weakly labeled* training data [15]. Another approach is to use active learning to select only the best training examples for human labeling [52]. A third approach is to utilize a mixture of both labeled and unlabeled data during training [12].

Improved learning algorithms for information extraction continue to be developed. Recently, a number of methods for improving HMMs have been proposed, including linear interpolating HMMs [11], maximum-entropy Markov models [36], and conditional random fields [30].

Existing biological knowledge can also be used to improve extraction performance. Currently we have only exploited dictionaries of known protein names. Using learning to revise initial human-written extraction rules has also been shown to improve performance [18]. One can imagine many other sources of external knowledge: global statistical properties of abstracts, existing interaction or pathway data, prior expectations for finding protein names, and dictionaries of near-miss negative examples of protein names. Filtering proposed interacting proteins by comparing their gene-expression data or examining their co-occurrences in other abstracts or web pages could also prove useful.

In the future, it will also be interesting to develop IE systems for extracting other associations with genes. A few examples include extracting information about post-translational modifications of proteins, identifying genes that are specifically involved with diseases, identifying genes that are co-regulated, extracting protein-drug interactions, protein-metabolite interactions and information about the dynamics and dependencies of these processes.

6 Acknowledgements

We would like to thank members of the Marcotte lab for helping to tag Medline abstracts. We would also like to thank Kristie Seymore for making the IE-tagging tool available. We would like to thank Lorraine Tanabe for kindly allowing us to use the ABGENE system. Many thanks to Kenichiro Fukuda for the KEX system, Mary Elaine Califf for the RAPIER system, Dayne Freitag and Nicholas Kushmerick for the BWI program, Thorsten Joachims for the SVM^{light} software, Eric Brill for his POS tagger, and to Ellen Riloff for the Sundance shallow parser. We also thank Christian Blaschke for the information about the hand-written rules. This work was supported in part by the National Science Foundation (IIS-0117308, ITR-0219061 and IIS-0325116), the Welch Foundation (F-1515), and the Texas Advanced Research Program.

References

[1] A. L. Berger, S. A. Della Pietra, V. J. Della Pietra, A maximum entropy approach to natural language processing, Computational Linguistics 22 (1) (1996) 39–71.

[2] D. M. Bikel, R. Schwartz, R. M. Weischedel, An algorithm that learns what's in a name, Machine Learning 34 (1999) 211–232.

[3] C. Blaschke, A. Valencia, Can bibliographic pointers for known biological data be found automatically? protein interactions as a case study, Comparative and Functional Genomics 2 (2001) 196–206.

[4] C. Blaschke, A. Valencia, The frame-based module of the Suiseki information extraction system, IEEE Intelligent Systems 17 (2002) 14–20.

[5] E. Brill, Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging, Computational Linguistics 21 (4) (1995) 543–565.

[6] M. E. Califf (Ed.), Papers from the 16th Natl. Conf. on Artificial Intelligence (AAAI-99) Workshop on Machine Learning for Information Extraction, AAAI Press, Orlando, FL, 1999.

[7] M. E. Califf, R. J. Mooney, Relational learning of pattern-match rules for information extraction, in: Proc. of 16th Natl. Conf. on Artificial Intelligence (AAAI-99), Orlando, FL, 1999, pp. 328–334.

[8] C. Cardie, Empirical methods in information extraction, AI Magazine 18 (4) (1997) 65–79.

[9] B. Cestnik, Estimating probabilities: A crucial task in machine learning, in: Proc. of 9th European Conf. on Artificial Intelligence, Stockholm, Sweden, 1990, pp. 147–149.

[10] C. Charras, T. Lecroq, Sequence comparison [12], Laboratoire d'Informatique de Rouen et Atelier Biologie Informatique Statistique Socio-Linguistique, Université de Rouen, France (1998).

[11] N. Collier, C. No, J. Tsujii, Extracting the names of genes and gene products with a Hidden Markov Model, in: Proc. of 18th Intl. Conf. on Computational Linguistics, Saarbruken, Germany, 2000, pp. 201–207.

[12] M. Collins, Y. Singer, Unsupervised models for named entity classification, in: Proc. of the Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99), University of Maryland, 1999.

[13] M. J. Collins, Ranking algorithms for named-entity extraction: Boosting and the voted perceptron, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), philadelphia, 2002, pp. 489–496.

[12] `http://www-igm.univ-mlv.fr/~lecroq/seqcomp`

[14] J. Cowie, W. Lehnert, Information extraction, Communications of the Association for Computing Machinery 39 (1) (1996) 80–91.

[15] M. Craven, J. Kumlien, Constructing biological knowledge bases by extracting information from text sources, in: Proc. of the 7th Intl. Conf. on Intelligent Systems for Molecular Biology, Heidelberg, Germany, 1999, pp. 77–86.

[16] J. Darroch, D. Ratchliff, Generalized iterative scaling for log-linear models, The Annals of Mathematical Statistics 43 (5) (1972) 1470–1480.

[17] R. O. Duda, P. E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[18] T. Eliassi-Rad, J. Shavlik, A theory-refinement approach to information extraction, in: Proc. of 18th Intl. Conf. on Machine Learning (ICML-2001), 2001.

[19] D. Freitag, N. Kushmerick, Boosted wrapper induction, in: Proc. of 17th Natl. Conf. on Artificial Intelligence (AAAI-2000), AAAI Press / The MIT Press, Austin, TX, 2000, pp. 577–583.

[20] Y. Freund, R. E. Schapire, Experiments with a new boosting algorithm, in: L. Saitta (Ed.), Proc. of 13th Intl. Conf. on Machine Learning (ICML-96), Morgan Kaufmann, 1996.

[21] C. Friedman, P. Kra, H. Yu, M. Krauthammer, A. Rzhetsky, GENIES: A natural-language processing system for the extraction of molecular pathways from journal articles, Bioinformatics 17 S74–S82, supplement 1.

[22] K. Fukuda, T. Tsunoda, A. Tamura, T. Takagi, Information extraction: Identifying protein names from biological papers, in: Proc. of the 3rd Pacific Symp. on Biocomputing, 1998, pp. 707–718.

[23] D. Gusfield, Algorithms on Strings, Trees and Sequences, Cambridge University Press, New York, 1997.

[24] U. Hahn, M. Romacker, S. Schulz, Creating knowledge repositories from biomedical reports: The MEDSYNDIKATE text mining system, in: Proc. of the 7th Pacific Symp. on Biocomputing, 2002, pp. 338–349.

[25] K. Humphreys, G. Demetriou, R. Geizauskas, Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structure, in: Proc. of the 5th Pacific Symp. on Biocomputing, 2000, pp. 502–513.

[26] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in: Proc. of 10th European Conf. on Machine Learning, Springer-Verlag, Berlin, 1998, pp. 137–142.

[27] M. Kubat, R. C. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Machine Learning 30 (2–3) (1998) 195–215.
URL `citeseer.nj.nec.com/kubat98machine.html`

[28] T. Kudoh, Y. Matsumoto, Use of support vector learning for chunk identification, in: Proc. of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000, pp. 142–144.

[29] N. Kushmerick, D. S. Weld, R. B. Doorenbos, Wrapper induction for information extraction, in: Proc. of 15th Intl. Joint Conf. on Artificial Intelligence (IJCAI-97), Nagoya, Japan, 1997, pp. 729–735.

[30] J. Lafferty, A. McCallum, F. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proc. of 18th Intl. Conf. on Machine Learning (ICML-2001), 2001.

[31] E. S. Lander, *et al.*, Initial sequencing and analysis of the human genome, Nature Feb 15;409(6822) (2001) 860–921.

[32] J. E. Leonard, J. B. Colombe, J. L. Levy, Finding relevant references to genes and proteins in medline using a Bayesian approach, Bioinformatics 18 (11) (2002) 1515–1522.

[33] V. I. Levenshtein, Binary codes capable of correcting insertions and reversals, Soviet Physics Doklady 10 (8) (1966) 707–710.

[34] E. Marcotte, I. Xenarios, D. Eisenberg, Mining literature for protein-protein interactions, Bioinformatics Apr;17(4) (2001) 359–363.

[35] M. Marcus, B. Santorini, M. A. Marcinkiewicz, Building a large annotated corpus of English: The Penn treebank, Computational Linguistics 19 (2) (1993) 313–330.

[36] A. McCallum, D. Freitag, F. Pereira, Maximum entropy Markov models for information extraction and segmentation, in: Proc. of 17th Intl. Conf. on Machine Learning (ICML-2000), Stanford, CA, 2000.

[37] National Institute of Standards and Technology, ACE - Automatic Content Extraction, http://www.nist.gov/speech/tests/ace/.

[38] J. Park, H. S. Kim, J. J. Kim, Bidirectional incremental parsing for automatic pathway identification with combinatory categorial grammar, in: Proc. of the 6th Pacific Symp. on Biocomputing, 2001, pp. 396–407.

[39] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo,CA, 1988.

[40] C. Perez-Iratxeta, P. Bork, M. A. Andrade, Association of genes to genetically inherited diseases using data mining, Nature Genetics 31 (3) (2002) 316–319.

[41] D. Proux, F. Rechenmann, L. Julliard, A pragmatic information extraction strategy for gathering data on genetic interactions, in: Proc. of the 9th Intl. Conf. on Intelligent Systems for Molecular Biology, 2000, pp. 279–85.

[42] D. Proux, F. Rechenmann, L. Julliard, V. Pillet, B. Jacq, Detecting gene symbols and names in biological texts: A first step toward pertinent information extraction, Genome Informatics 9 (1998) 72–80, GIW '98.

[43] J. Pustejovsky, J. Castano, J. Zhang, M. Kotecki, B. Cochran, Robust relational parsing over biomedical literature: Extracting inhibit relations, in: Proc. of the 7th Pacific Symp. on Biocomputing, 2002, pp. 362–373.

[44] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. of the IEEE 77 (2) (1989) 257–286.

[45] L. A. Ramshaw, M. P. Marcus, Text chunking using transformation-based learning, in: Proc. of 3rd Workshop on Very Large Corpora, 1995.

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

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

[48] T. C. Rindflesch, L. Tanabe, J. N. Weinstein, L. Hunter, EDGAR: Extraction of drugs, genes, and relations from the biomedical literature, in: Proc. of the 5th Pacific Symp. on Biocomputing, 2000, pp. 515–524.

[49] D. Roth, A. van den Bosch (Eds.), Proc. of 6th Conf. on Natural Language Learning, Association for Computational Linguistics, Taipei, Taiwan, 2002.

[50] L. Tanabe, W. J. Wilbur, Tagging gene and protein names in biomedical text, Bioinformatics 18 (8) (2002) 1124–1132.

[51] J. Thomas, D. Milward, C. Ouzounis, S. Pulman, M. Carol, Automatic extraction of protein interactions from scientific abstracts, in: Proc. of the 5th Pacific Symp. on Biocomputing, 2000, pp. 541–553.

[52] C. A. Thompson, M. E. Califf, R. J. Mooney, Active learning for natural language parsing and information extraction, in: Proc. of 16th Intl. Conf. on Machine Learning (ICML-99), Bled, Slovenia, 1999, pp. 406–414.

[53] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, Berlin, 1995.

[54] V. N. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.

[55] J. C. Venter, *et al.*, The sequence of the human genome, Science Feb 16;291(5507) (2001) 1304–1351.

[56] I. Xenarios, E. Fernandez, L. Salwinski, X. J. Duan, M. J. Thompson, E. M. Marcotte, D. Eisenberg, DIP: The database of interacting proteins: 2001 update, Nucleic Acids Research 29 (1) (2001) 239–241.