Submitted for publication

Learning to Extract Proteins and their Interactions from Medline Abstracts

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

[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 hidden Markov models 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 more accurately 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 have focused on the manual development of IE systems for extracting information from biomedical literature [1–7]. Unfortunately, manual engineering of information extraction (IE) systems for particular applications is a tedious and time-consuming process [8]. 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 [9,10]. 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 information. Next, the IE systems induced from this supervised data are used to

* Corresponding Author.
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 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 the Welch Foundation (F-1515), the National Science Foundation (ITR-0219061), and the Texas Advanced Research Program.
[2] Supported by grant IIS-0117308 from the National Science Foundation.
[3] Supported by an MCD Fellowship from the University of Texas at Austin.

extract new information from novel test documents. Some projects on extracting information from biomedical literature have also employed such learning techniques [11–18].

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 [19,20], 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 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) [21], boosted wrapper induction (BWI) [22], memory-based learning (MBL) [23], transformation-based learning (TBL) [24], support vector machines (SVMs) [25], and hidden Markov models (HMMs) [26,14]. 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[4] 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 tags. 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 [27]) 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 generated by scanning approximately 5,000 abstracts for sentences that mention at least two genes/proteins that do not interact.

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. Manual examination of many abstracts revealed several ambiguities, such as whether the organism names should be tagged (e.g. `<prot> human delta catenin </prot>` or `human <prot> delta catenin </prot>`), whether punctuation should be tagged (e.g. `( <prot> LIGHT </prot> )` or `<prot> ( LIGHT ) </prot>`), and whether generic protein family names should be tagged (e.g. `<prot> armadillo protein p0071 </prot>` or `armadillo protein`

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

```
PMID -- 9367879
TI -- A <prot> c - Cbl </prot> yeast two hybrid screen
reveals interactions with 14 - 3 - 3 isoforms and cytoskeletal
components .
PG -- 46 - 50 AB - The protein product of <p1 pair=1> <p1
pair=2> <p1 pair=3> <prot> c - cbl </prot> </p1> </p1> </p1>
proto - oncogene is known to interact with several proteins ,
including <p2 pair=1> <prot> Grb2 </prot> </p2> , <p2 pair=2>
<prot> Crk </prot> </p2> and <p2 pair=3> <prot> PI3 kinase
</prot> </p2> , and is thought to regulate signalling by many
cell surface receptors .
The precise function of <prot> c - Cbl </prot> in these pathways
is not clear , although a genetic analysis in Caenorhabditis
elegans suggests that <p1 pair=4> <prot> c - Cbl </prot>
</p1> is a negative regulator of the <p2 pair=4> <prot> <prot>
epidermal growth factor </prot> receptor </prot> </p2> . Here we
describe a yeast two hybrid screen performed with <prot> c - Cbl
</prot> in an attempt to further elucidate its role in signal
transduction . The screen identified interactions involving
<p1 pair=5> <p1 pair=6> <prot> c - Cbl </prot> </p1> </p1> and
two 14 - 3 - 3 isoforms , <p2 pair=5> <prot> cytokeratin 18
</prot> </p2> , human unconventional myosin IC , and a recently
identified SH3 domain containing protein , <p2 pair=6> <prot>
SH3 P17 </prot> </p2> . We have used the yeast two hybrid assay
to localise regions of <prot> c - Cbl </prot> 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 tag PROT indicates protein name and the tags P1 and P2 having the same value for the attribute PAIR indicate interaction between the proteins.

`<prot> p0071 </prot>`). 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. `<prot> <prot> GITR </prot> ligand </prot>`)
(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 <p1 pair=1> <prot> <prot> GITR </prot> </p1> ligand </prot> ( <p2 pair=1> <prot> hGITRL <prot> </p2> )`)

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 [26,28]. 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[5].
(2) The file `feb2002-tables.tar.gz`, downloaded from the Gene Ontology Database[6].

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.

[5] URL: `http://us.expasy.org/sprot/hpi/hpi_ftp.html`
[6] 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.

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 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. As 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 [21] 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 preceding the filler, and all words 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

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

the best rule in terms of information gain produces no spurious fillers when matched against the training examples. The best generalized 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) [22] 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 [30], 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 Memory-based Learning

In the remaining methods, we approach protein name identification from a slightly different angle. 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, a 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 [31,28].

Notice that this approach does not work when a protein name is part of another protein name. But since such protein names are rare (127 out of 5,206 in our corpus, or 2.44%), we are going to ignore this possibility, and assume that some protein names will always be missed whenever name nesting occurs.

We tested the efficacy of this approach using several traditional classification methods. The first is k-nearest neighbor (k-NN) or *memory-based learning*, in which classification is done by extrapolation from the k most similar training examples. We used version 4.2 of TiMBL [7], which has been successfully applied to several text chunking tasks [32,33].

For each word in the corpus, we formed a feature vector, consisting of the word itself, the previous N words, 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 (Section 3.1.1) for all $2N+1$ words. We ignored capitalization when preparing the feature vectors to avoid sparsity. To calculate the similarity

[7] URL: `http://ilk.kub.nl/software.html`

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

between two feature vectors, we used the overlap metric weighted by chi-squared statistics. Good values for k and N were determined empirically (see Section 3.2.2).

3.1.5 Support Vector Machines

Support Vector Machines (SVMs) are one of the most recently developed classification methods [34]. 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 [35].

Assume that all training examples consist of a vector of m 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 proved 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.

In our experiments with SVMs, we used the same set of features as in Section 3.1.4, with N set to 2. Inspired by the text chunking algorithm presented in [36], 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. Since numerical values were needed, each word or tag in each position was a separate binary feature. By taking all words and tags appearing in the training data as features, the dimension of feature vectors became as large as 74,487. For each extracted string, 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 [35], 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.

3.1.6 Transformation-based Learning

Transformation-based learning (TBL) is a rule-based, error-driven learning technique that has been applied to a number of natural language problems

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

[24]. It is the basis of the widely-used Brill's POS tagger. A summary of the learning process is as follows. First, a manually annotated corpus is passed through a lexical rule learner, which learns lexical rules that assign initial tags by looking at the lexical features of each word, such as whether the word is capitalized, and what the last three letters of the word are. After that, the lexical rules are applied to the same corpus with all annotations removed. Then transformation rules are learned by comparing the output of initial tagging to the correct annotations. Contextual information is taken into account when generating transformation rules. An example rule is "change the tag from NIL to PROT if the following word is *protein*." [10] The goal of transformation rules is to make the annotations better resemble the correct ones. The learner stops when no more transformations that result in a certain amount of error reduction can be found. During testing, new text is annotated by first applying the lexical rules, and then applying each of the transformation rules in order.

In our first experiment with TBL, we applied the lexical rule and contextual rule learners to our training set, in order to see if the learners were able to learn rules from scratch. In our second experiment, we applied the dictionary-based tagger (Section 3.1.1) to the training set, and compared the results to the correct annotations, from which contextual rules were induced.

3.1.7 Hidden Markov Models

Hidden Markov Models (HMMs) can be viewed as stochastic finite state automata defined in terms of a set of states and transitions between them. Each state is associated with two probability distributions: one over the emission of various tokens, and the other over the transitions to other states in the model. These distributions can be learned from training data using maximum-likelihood estimates when the states are known, or the Baum-Welch procedure, if the states are unknown [37].

HMMs have long been used in speech recognition and part-of-speech tagging [38], and they have also proved to be highly effective in information extraction tasks, such as name entity recognition [26], or template filling [39]. Tasks from biomedical domain have also benefited from the use of HMM based supervised learning. In one approach to protein-name extraction, an HMM model is learned based on word features such as capitalization, Greek letters, digits and symbols, without relying on other types of information such as POS tags, handcrafted rules, or dictionaries [13]. Another work has addressed the task of identifying subcellular structures in which proteins are located by learning HMM models with different levels of incorporated knowledge, from an HMM based on words only, to a model using part-of-speech information,

[10] If the tag is NIL, then the current word is not recognized as part of a protein name. The tag PROT carries the opposite meaning.

and finally to a third model which relies on syntactic information obtained from a shallow parser [14]. The model that uses words and part-of-speech information is also the model used in our experiments. It involves creating two sub-models:

- A *positive model* in which states correspond to either a part-of-speech alone, or to a part-of-speech annotated as protein. In the recognition phase, words are tagged as being part of a protein name only if they are emitted by a protein annotated state.
- A *null model* containing a different state for each part-of-speech.

These two HMMs are intended to recognize sentences containing protein names (the positive model) and sentences without any protein names (the null model). Consequently, the positive model is trained on sentences tagged with protein names, while the null model is trained on the remaining sentences. Both models are fully connected. In the extraction phase, a sentence is deemed as positive (i.e. containing protein names) only if the likelihood of emission by the positive model is greater than the likelihood of emission by the null model. In order to extract the protein names in a positive sentence, the system identifies the maximal contiguous sequences of words that are emitted by protein annotated states from the Viterbi path of the sentence in the positive model. In a variant of the system, we used the output of the dictionary-based tagger (Section 3.1.1) to replace the POS tag of tokens inside a tagged sequence with a separate tag.

3.1.8 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 [1]. 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.

We also tested ABGENE, introduced in [40]. Similar to the method presented in Section 3.1.6, ABGENE uses a transformation-based tagger to produce initial tagging. Then it uses 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.

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 [41]. 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) [8].

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. For the k-NN tagger (Section 3.1.4), we determined the best parameter settings by testing on every combination of $k = 1, 3, 5$ and $N = 1, 2, 3$ using only words as features. Then we used the combination of k and N that gave the best F-measure in the remaining experiments on k-NN. For systems that output confidences that allow trading-off precision and recall (i.e. BWI, k-NN, SVM and HMM), results are presented for the maximum achievable recall.

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	76.77%	10.22%	18.04%
part-of-speech	72.95%	11.08%	19.24%
dictionary-based tagger	74.23%	12.45%	**21.32%**
BWI (300 iterations, 2 lookaheads, max. recall)			
words only	72.97%	11.56%	19.96%
dictionary-based tagger	69.50%	24.03%	**35.71%**
k-NN (words only, max. recall)			
$k = 1, N = 1$	37.18%	33.67%	35.34%
$k = 1, N = 2$	39.21%	36.73%	**37.93%**
$k = 1, N = 3$	37.34%	37.17%	37.25%
$k = 3, N = 1$	23.90%	8.99%	13.07%
$k = 3, N = 2$	33.43%	19.80%	24.87%
$k = 3, N = 3$	34.96%	24.30%	28.67%
$k = 5, N = 1$	13.54%	0.25%	0.49%
$k = 5, N = 2$	22.45%	7.03%	10.71%
$k = 5, N = 3$	29.82%	14.00%	19.06%
k-NN ($k = 1, N = 2$, max. recall)			
part-of-speech	34.79%	40.86%	37.58%
dictionary-based tagger	47.22%	47.94%	**47.58%**
SVM ($N = 2$, max. recall)			
preceding class labels	68.92%	19.77%	30.72%
preceding class labels and part-of-speech	69.97%	19.78%	30.85%
preceding class labels and dictionary-based tagger	64.86%	45.66%	**53.59%**
dictionary-based tagger but no preceding class labels	60.75%	47.31%	53.20%
HMM (max. recall)			
part-of-speech	49.21%	25.93%	33.96%
dictionary-based tagger	51.24%	33.73%	40.68%
part-of-speech and dictionary-based tagger	60.29%	39.95%	**48.05%**
KEX	14.68%	31.83%	**20.09%**
Abgene	32.39%	45.87%	**37.97%**

Table 1
Performance of protein taggers in different settings

For ease of comparison, we show recall-precision curves in Figure 4, 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.

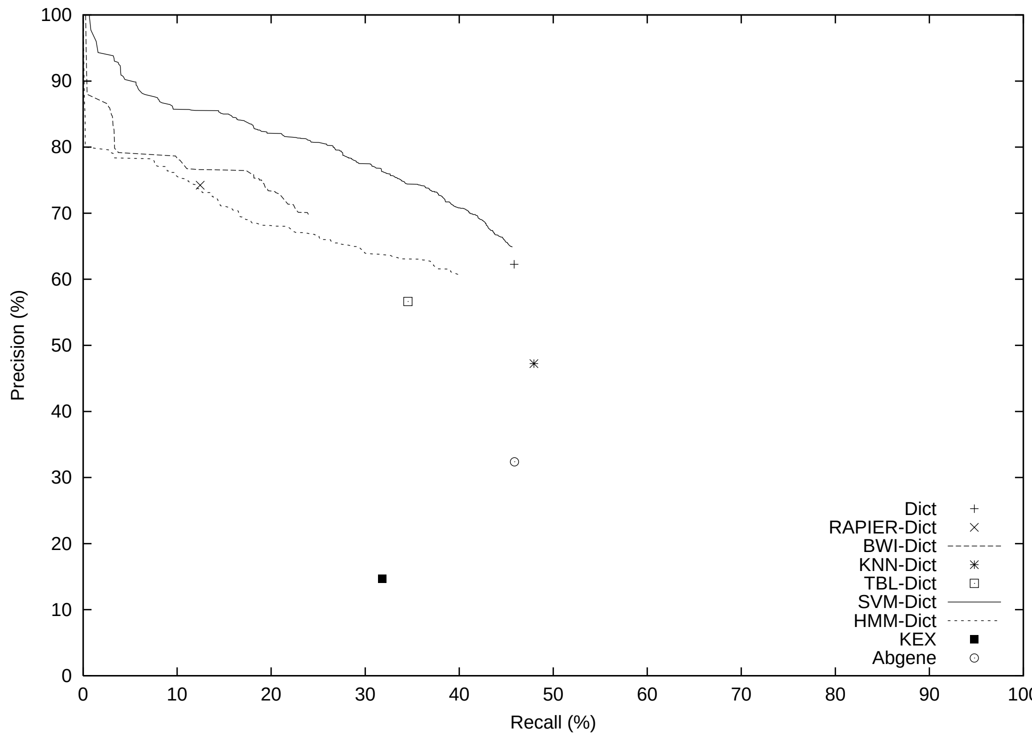

Fig. 4. Precision-recall curves for protein taggers in their best setting

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 ($p > 0.05$). An exception is for HMM, where the F-measure increases significantly when both the POS tags and the output of the dictionary-based tagger are available. This is because the inclusion of POS tags results in a model with more states, allowing for more accurate modeling of the data.

While the dictionary-based tagger barely improves 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 k-NN and HMM, while for BWI and SVM, it hurts the precision slightly, but this is outweighed by a larger gain in recall. For SVM, the improvement in recall is so great that SVM coupled with the dictionary-based tagger gives the best overall F-measure.

While none of the learning methods achieve a significant improvement over the dictionary-based tagger in terms of F-measure, several of them can produce much higher precision. In particular, SVM is able to achieve arbitrarily high precision by adjusting the level of recall. This is because the extraction confidence seems to truly reflect the probability of correctness. Since high precision is needed to extract accurate knowledge from text, this is a significant contribution.

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.

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) [42] 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 [12,14,15,2–6,43]. 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.

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

```
<interactor> SHPTP2 </interactor> <interfiller> interacts with
another signaling protein , </interfiller> <interactee> Grb7
</interactee> .
```

Fig. 5. Interactor, interactee and interfiller

tagged entities participating in a relationship, the text fragment between them is called the interfiller (see Figure 5). 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 into different role-specific slots. For protein interactions, we named the roles *interactor* and *interactee* (see Figure 5). We then assume that all interacting proteins appear in the same sentence and extract the related pairs using the following heuristics. (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 occuring interactee in the segment. (3) 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.* [3,44] 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.

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

Fig. 6. Sample protein-extraction rule learned by ELCS.

ELCS' rule representation is similar to that in [3,44], except that it currently does not use POS tags, but allows disjunctions of words. Figure 6 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 satisifes the word constraints in the given order and respects the respective word gaps.

Sentences in the training data that contain interacting proteins are called positive sentences and others are called negative sentences. ELCS induces rules using a bottom-up approach. It starts with positive sentences and repeatedly generalizes them to form rules until the rules becomes overly general and start matching negative sentences. Note that a positive sentence 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 positive sentences. To do this we replicate positive 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.

Rule induction starts with maximally specific rules for each positive sentence which contain all the words in the sentence with zero-length word gaps. We have developed two methods for generalizing rules, one that does not use disjunction and one that does. A simple way to produce a non-disjunctive 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 [45,46]. 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 orignal two rules. Figure 7 shows a sample generalization of two sentences.

Sentence 1: `The <p1> retinoblastoma </p1> protein binds to <p2> RIZ </p2> , a zinc - finger protein that shares an epitope with the adenovirus E1A protein .`
Sentence 2: `The present study has shown that cell surface <p1 pair=1> calreticulin </p1> binds to the <p2 pair=1> B beta chain of fibrinogen </p2> mediating its mitogenic activity .`
Generalization using LCS: The (7) PROTEINNAME (1) binds (0) to (1) PROTEINNAME (15) .

Fig. 7. Sample LCS generalization of two sentences with P1 and P2 protein tags.

Sentence 1: `The <p1> retinoblastoma </p1> protein binds to <p2> RIZ </p2> , a zinc - finger protein that shares an epitope with the adenovirus E1A protein .`
Sentence 2: `The present study has shown that cell surface <p1 pair=1> calreticulin </p1> binds to the <p2 pair=1> B beta chain of fibrinogen </p2> mediating its mitogenic activity .`
Generalization using edit distance: The (7) PROTEINNAME (1) binds (0) to (1) PROTEINNAME (11) [mediating | the] (0) [its | adenovirus] (0) [mitogenic | E1A] (0) [activity | protein] (0) .

Fig. 8. Sample edit-distance generalization of two sentences with P1 and P2 protein tags.

Our second approach to generalization uses *edit distance* (ED) [46] and creates more specific rules that contain disjunctive constraints. The most common edit distance is Levenshtein distance [47], 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. Figure 8 shows an example of edit-distance generalization.

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 of all the positive sentences without covering many negative ones. Specifically, we use a version of the algorithm used in GOLEM [48,49]. GOLEM first generates a good seed rule by generalizing random pairs of positive examples. Then it repeatedly generalizes this rule as long as it continues to cover more positives without covering too many negatives. It returns the rule thus learned, removes all of the positive examples it covers, and then continues learning rules until all of the positive examples are covered.

In order to prevent over-fitting, we allow learned rules to cover a small number interactions (0) between (4) PROTEINNAME (0) and (4) PROTEINNAME (16) .

PROTEINNAME (3) - (6) PROTEINNAME (21) in (5) cell (0) lines (16) .

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

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

[We | Structure | we | strong | determinants | Interactions | consisting | consists] (0) [demonstrate | of | interaction | for] (4) PROTEINNAME (12) [binds | bound | protein | homology | assembly | but] (19) [to | (| , | designated | specific] (11) PROTEINNAME (19) .

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

Fig. 9. Some example rules learned by ELCS; the first three were learned using LCS generalization and the next three using edit distance generalization.

of negative examples. A rule is kept as long as $\frac{p-n}{p+n} > C$, where p and n are, respectively, the number of positive and negative examples covered by the rule, and C is a parameter of the system. The greater the value of C, the more induced rules tend towards high precision but low recall. A similar approach is used in RIPPER [50] and Rapier [21].

The generalizations obtained using either the LCS or ED method may result in rules that 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. However, constraining learned rules to contain two protein names is a useful bias. Therefore, we divide each of the training sentence 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 9 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 two proteins within a sentence are considered when identifying interactions (this constraint is satisfied by all interactions in our corpus). We also compared our systems with Blaschke *et al.*'s manually-written rules [44]. Since these rules use POS tags, we also 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].

Our current experiments only evaluate the performance of interaction extraction, assuming all protein names have already been correctly tagged. Results with automatically-tagged proteins will be presented in the final version of the paper. As in Section 3.2.1, performance is evaluated using ten-fold cross validation and measuring recall and precision. We consider an extracted interaction correct if and only if the names and positions of both proteins exactly match human-tagged interactions. Note that this is the strictest measure of accuracy.

4.2.2 Quantitative Results

Figure 10 shows recall-precision results for protein-interaction extraction when tested on abstracts that have been manually tagged for protein names. We plotted a precision-recall curve for BWI by utilizing its extraction confidences. Since RAPIER and human-written rules do not produce confidences, only a single recall-precision point is shown. For ELCS, we show the point which gives the maximum F-measure, found by varying the overfitting parameter C.

4.2.3 Discussion of Results

BWI gives varying degrees of high precision, but recall is generally quite low. RAPIER also gives relatively high precison but low recall. ELCS tends to give higher recall with only a modest decrease in precision compared to BWI and RAPIER. In contrast, human-written rules with POS tags achieve high recall but quite low precision. The human-written rules with typical words substituted for POS tags gives somewhat higher precision but with much less recall.

These results generally demonstrate that machine learning out-performs human-

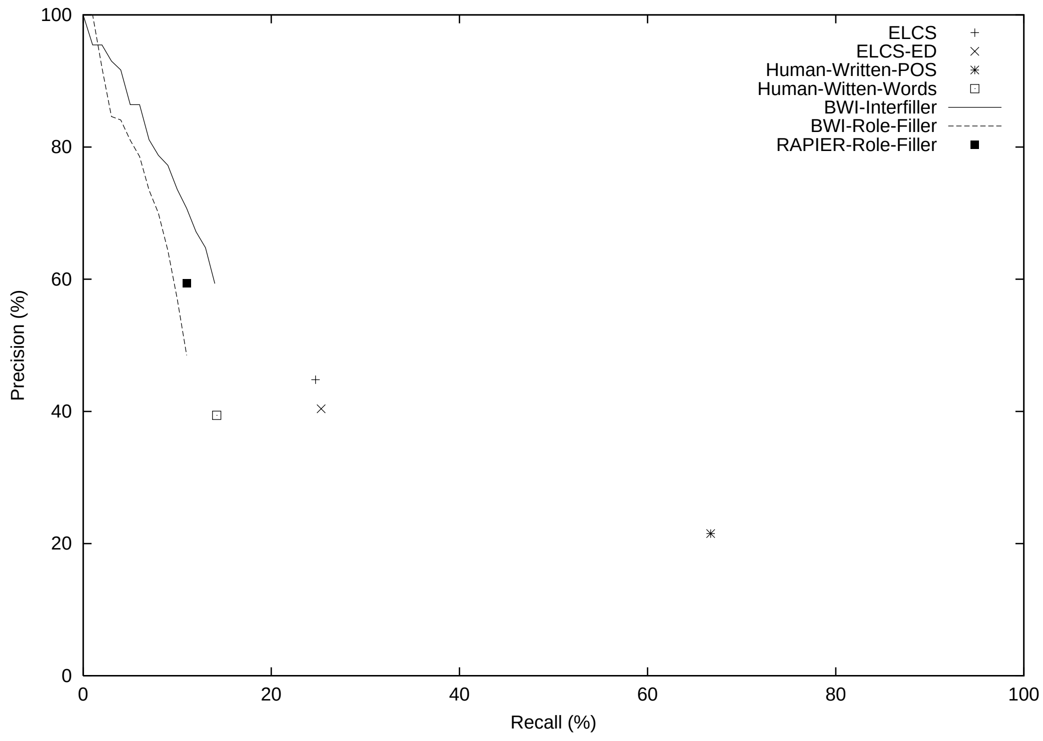

Fig. 10. Precision-recall graphs for protein interaction extraction.

written rules. The human-written rules with POS tags give higher recall but very low precision. When the human-written rules are made more precise by replacing the POS tags by specific key words, as in [3], then the machine learning systems clearly out-perform the human-written rules in terms of both precision and recall. 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.

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 an SVM-based learning method that exploits a generalized protein-name dictionary. Other learning methods also perform fairly well; however, no method achieves greater than 50% recall. For extracting protein interactions, we found that several methods for learning extraction rules generally out-perform hand-written rules, particularly with respect to precision. Token classification methods like k-NN, SVM, and TBL 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 [12]. Another approach is to use active learning to select only the best training examples for human labeling [51]. A third approach is to utilize a mixture of both labeled and unlabeled data during training [52].

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 [13], maximum-entropy HMMs [53], and conditional random fields [54].

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 [15]. 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-occurences 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, and Soumya Ray and Mark Craven for their HMM program. 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.

References

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

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

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

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

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

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

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

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

[10] M. E. Califf (Ed.), Papers from the Sixteenth National Conference on Artificial Intelligence (AAAI-99) Workshop on Machine Learning for Information Extraction, AAAI Press, Orlando, FL, 1999.

[11] 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.

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

[13] N. Collier, C. No, J. Tsujii, Extracting the names of genes and gene products with a hidden markov model, in: Proceedings of the Eighteenth International Conference on Computational Linguistics, Saarbruken, Germany, 2000, pp. 201–207.

[14] S. Ray, M. Craven, Representing sentence structure in hidden Markov models for information extraction, in: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001), Seattle, WA, 2001, pp. 1273–1279.

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

[16] 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.

[17] 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.

[18] 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.

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

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

[21] M. E. Califf, R. J. Mooney, Relational learning of pattern-match rules for information extraction, in: Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), Orlando, FL, 1999, pp. 328–334.

[22] D. Freitag, N. Kushmerick, Boosted wrapper induction, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), AAAI Press / The MIT Press, Austin, TX, 2000, pp. 577–583.

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

[24] 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.

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

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

[27] 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.

[28] D. Roth, A. van den Bosch (Eds.), Proc. of the Sixth Conference on Natural Language Learning, Association for Computational Linguistics, Taipei, Taiwan, 2002.

[29] N. Kushmerick, D. S. Weld, R. B. Doorenbos, Wrapper induction for information extraction, in: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, Japan, 1997, pp. 729–735.

[30] Y. Freund, R. E. Schapire, Experiments with a new boosting algorithm, in: L. Saitta (Ed.), Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96), Morgan Kaufmann, 1996.

[31] L. A. Ramshaw, M. P. Marcus, Text chunking using transformation-based learning, in: Proceedings of the Third Workshop on Very Large Corpora, 1995.

[32] W. Daelemans, S. Buchholz, J. Veenstra, Memory-based shallow parsing, in: Proceedings of CoNLL-1999, Bergen, Norway, 1999.

[33] E. T. K. Sang, Memory-based shallow parsing, Journal of Machine Learning Research 2 (2002) 559–594.

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

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

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

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

[38] J. Kupiec, Robust part-of-speech tagging using a hidden markov model, Computer Speech and Language 6 (1992) 225–242.

[39] D. Freitag, A. McCallum, Information extraction with HMM structures learned by stochastic optimization, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), AAAI Press / The MIT Press, Austin, TX, 2000.

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

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

[42] N. I. of Standards, Technology, ACE - Automatic Content Extraction, http://www.nist.gov/speech/tests/ace/.

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

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

[45] C. Charras, T. Lecroq, Sequence comparison[11], Laboratoire d'Informatique de Rouen et Atelier Biologie Informatique Statistique Socio-Linguistique, Université de Rouen, France (1998).

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

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

[48] S. Muggleton, C. Feng, Efficient induction of logic programs, in: Proceedings of the First Conference on Algorithmic Learning Theory, Ohmsha, Tokyo, Japan, 1990.

[49] N. Lavrac, S. Dzeroski, Inductive Logic Programming: Techniques and Applications, Ellis Horwood, 1994.

[50] W. W. Cohen, Fast effective rule induction, in: Proceedings of the Twelfth International Conference on Machine Learning (ICML-95), San Francisco, CA, 1995, pp. 115–123.

[51] C. A. Thompson, M. E. Califf, R. J. Mooney, Active learning for natural language parsing and information extraction, in: Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99), Bled, Slovenia, 1999, pp. 406–414.

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

[53] A. McCallum, D. Freitag, F. Pereira, Maximum entropy Markov models for information extraction and segmentation, in: Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000), Stanford, CA, 2000.

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

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