To Appear in *Statistical Relational Learning* Lise Getoor and Ben Taskar (Eds.)
forthcoming book, 2005

1 Statistical Relational Learning for Natural Language Information Extraction

Razvan C. Bunescu
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233
razvan@cs.utexas.edu
http://www.cs.utexas.edu/∼razvan

Raymond J. Mooney
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233
mooney@cs.utexas.edu
http://www.cs.utexas.edu/∼mooney

1.1 Introduction

Understanding natural language presents many challenging problems that lend themselves to statistical relational learning (SRL). Historically, both logical and probabilistic methods have found wide application in natural language processing (NLP). NLP inevitably involves reasoning about an arbitrary number of entities (people, places, and things) that have an unbounded set of complex relationships between them. Representing and reasoning about unbounded sets of entities and relations has generally been considered a strength of predicate logic. However, NLP also requires integrating uncertain evidence from a variety of sources in order to resolve numerous syntactic and semantic ambiguities. Effectively integrating multiple sources of uncertain evidence has generally been considered a strength of Bayesian probabilistic methods and graphical models. Consequently, NLP problems are particularly suited for SRL methods that combine the strengths of first-order predicate logic and probabilistic graphical models. In this article, we review our recent work (Bunescu and Mooney [2004]) on using Relational Markov Networks

(RMNs) (Taskar et al. [2002]) for information extraction, the problem of identifying phrases in natural language text that refer to specific types of entities (Cardie [1997]). We use the expressive power of RMNs to represent and reason about several specific relationships between candidate entities and thereby collectively identify the appropriate set of phrases to extract. We present experiments on learning to extract protein names from biomedical text, which demonstrate the advantage of this approach over existing IE methods.

The remainder of the article is organized as follows. In Section 1.2, we review the history of logical and probabilistic approaches to NLP, and discuss the unique suitability of SRL for NLP. Section 1.3 introduces the problem of information extraction, followed by Section 1.4, where we summarize our work on collective information extraction using RMNs. In Section 1.5, we examine challenging problems for future research on SRL for NLP. In Section 1.6, we present our final conclusions.

1.2 Background on Natural Language Processing

Early research in NLP focused on symbolic techniques in which the knowledge required for understanding and generating language consisted of manually written production rules, semantic networks, and/or axioms in predicate logic (Allen [1987]). The semantic analysis of language was a particular focus of NLP research in the 1970's, with researchers exploring tasks ranging from responding to commands and answering questions in a micro-world (Winograd [1972]) to answering database queries (Woods [1977]) and understanding short stories (Schank and Riesbeck [1981]). These early systems could perform impressive semantic interpretation and inference when understanding particular sentences or stories; however, they tended to require tedious amounts of application-specific knowledge-engineering and were therefore quite brittle and not easily extended to new texts or new applications.

Disenchantment with the knowledge-engineering requirements and brittleness of symbolic, manually-developed NLP systems grew. Meanwhile, researchers in speech recognition started to obtain promising results using statistical methods trained on large annotated corpora (Jelinek [1976]). Eventually, statistical methods came to dominate speech recognition (Jelinek [1998]), and this development began to motivate the application of similar methods to other aspects of NLP, such as part-of-speech (POS) tagging (Church [1988]).

During the early 1990's, research in computational linguistics underwent a dramatic paradigm shift. Statistical learning methods that automatically acquire knowledge for language processing from empirical data largely supplanted systems based on human knowledge engineering (Hirschberg [1998], Manning and Schütze [1999]). However, in order to avoid the difficult problems of detailed semantic interpretation, NLP research focused on building robust systems for simpler tasks, such as POS tagging, syntactic parsing, word-sense disambiguation, and information extraction of specific types of entities.

Many of the methods used in statistical NLP are fundamentally SRL techniques since they perform some form of collective classification on unbounded length strings. Strings can be seen as simple instances of relational data where the individual items are characters, words, or tokens and the single relation "after" holds between adjacent items. Many NLP tasks, such as POS tagging, phrase chunking (Ramshaw and Marcus [1995]), and information extraction (e.g. named entity tagging), can be viewed as sequence labelling problems in which each word is assigned one of a small number of class labels. The label of each word typically depends on the labels of adjacent words in the sentence and collective inference must be performed to assign the overall most probable combination of labels to all of the words in the sentence. Statistical sequence models such as Hidden Markov Models (HMMs) (Rabiner [1989]) or Conditional Random Fields (CRFs) (Lafferty et al. [2001]) are used to model the data and some form of the Viterbi dynamic programming algorithm (Viterbi [1967]) is used to efficiently perform the collective classification. However, in order to develop systems that accurately and robustly perform natural language analysis, we believe that more advanced SRL methods are needed. In this paper, we explore the application of an alternative SRL method to the natural language task of information extraction. We introduce the task in the following section and then present our recent SRL approach.

1.3 Information Extraction

Information extraction (IE), locating references to specific types of items in natural language documents, is an important task with many practical applications. Typical examples include identifying various "named entities" such as names of people, companies, and locations. In this paper, we consider the IE task of identifying names of human proteins in abstracts of biomedical journal articles. Figure 1.1 shows part of a sample abstract highlighting the protein names to be identified. This task is an important part of mining the scientific literature in order to build structured databases of existing biological knowledge. In particular, by mining 753,459 abstracts on the human organism from the Medline repository (http://www.ncbi.nlm.nih.gov/entrez/) we have extracted a database of 6,580 interactions among 3,737 human proteins. The details of this database have been published in the biological literature (Ramani et al. [2005]) and it is available on the web at `http://bioinformatics.icmb.utexas.edu/idserve`.

> `Production of nitric oxide ( NO ) in endothelial cells is regulated by direct interactions of` *endothelial nitric oxide synthase* `(` *eNOS* `) with effector proteins such as` *Ca2+ – calmodulin* `. Here we have ... identified a novel 34 kDa protein , termed` *NOSIP* `(` *eNOS interaction protein* `) , which avidly binds to the carboxyl terminal region of the` *eNOS* `oxygenase domain .`

Figure 1.1 Medline abstract with all protein names emphasized.

In the simplest case, protein name identification can be treated as a sequence labeling problem in which each word (token) in the text is classified as either part of a protein name or not part of a protein name. As long as protein names are not immediately contiguous (a constraint consistently satisfied in the more than 1,000 human-annotated abstracts we have examined), this labeling allows immediate recovery of all substrings constituting protein names. However, in practice, a larger set of word labels can result in more accurate extraction. In particular, we found that five word labels: Begin (the first word in a multi-word name), End (the last word in a multi-word name), Inside (an internal word in a multi-word name), Single (a word corresponding to a single-word name), and Other (a word that is not part of a name), gave the best empirical results by creating word classes with the most easily captured regularities.

In a recent follow-up to previously published experiments comparing a wide variety of IE-learning methods (including HMM, SVM, MaxEnt, and rule-based methods) on the task of tagging references to human proteins in Medline abstracts (Bunescu et al. [2005]), CRFs were found to out-perform competing techniques (Ramani et al. [2005]). However, although CRFs capture the dependence between the labels of adjacent words, it does not adequately capture long-distance dependencies between potential extractions in different parts of a document. For example, in our protein-tagging task, repeated references to the same protein are common. If the context surrounding one occurrence of a phrase is very indicative of it being a protein, then this should also influence the tagging of another occurrence of the same phrase in a different context which is not typical of protein references. Consequently, more complex SRL methods that can capture such dependencies may result in more accurate information extraction. In the following section we show how RMNs can be used to model long-distance dependencies in the context of information extraction (for two recent alternative approaches, see the skip-chain CRFs introduced in (Sutton and McCallum [2004]) and the Gibbs sampling method from (Finkel et al. [2005])).

1.4 Collective Information Extraction with RMNs

In this section, we present our research on using RMNs to collectively extract all of the entities in a particular document. In particular, we have tested our approach on the difficult problem of identifying names of human proteins in biomedical journal abstracts. Unlike proteins in some other organisms (e.g. yeast), human proteins have no standardized nomenclature, making them particularly difficult to recognize amongst the variety of entity types referenced in biomedical text. One important source of potential evidence is the correlations between the labels of repeated phrases inside a document, as well as between acronyms and their corresponding long form. In both cases, the mentioned phrases tend to have the same entity label. For example, Figure 1.2 shows part of an abstract from Medline, an online database of biomedical articles. In this abstract, the protein referenced by "rpL22" is first introduced by its long name "ribosomal protein L22", followed by the short name "rpL22" between parentheses. The presence of the word "protein" is a very good indicator that the entire phrase "ribosomal protein L22" is a protein name. Also, "rpL22" is an acronym of "ribosomal protein L22", which increases the likelihood that it too is a protein name. The same name "rpL22" occurs later in the abstract in contexts which do not indicate so clearly the entity type, however we can use the fact that repetitions of the same name tend to have the same type inside the same document.

The control of human *ribosomal protein L22* (*rpL22*) to enter into the nucleolus
and its ability to be assembled into the ribosome is regulated by its sequence .
The nuclear import of *rpL22* depends on a classical nuclear localization signal
of four lysines at positions 13 - 16 . *RpL22* normally enters the nucleolus via
a compulsory sequence of KKYLKK (I - domain , positions 88 - 93) ... Once it
reaches the nucleolus , the question of whether *rpL22* is assembled into the
ribosome depends upon the presence of the N - domain .

Figure 1.2 Medline abstract with all protein names emphasized.

The capitalization pattern of the name itself is another useful indicator, nevertheless it is not sufficient by itself, as similar patterns are also used for other types of biological entities such as cell types or aminoacids. Therefore, correlations between the labels of repeated phrases, or between acronyms and their long form can provide additional useful information. Our intuition is that a method that could use this kind of information would show an increase in performance, especially when doing extraction from biomedical literature, where phenomena like repetitions and acronyms are pervasive. This type of document-level knowledge can be captured using Relational Markov Networks (RMNs), a version of undirected graphical models which have already been successfully used to improve the classification of hyper-linked web pages (Taskar et al. [2002]).

The rest of this section is organized as follows. In Sections 1.4.1 and 1.4.2 we describe the input to our named entity extractor in terms of a set of candidate entities and their features. Subsequent sections introduce the RMN framework for entity recognition (representation, inference and learning), ending with experimental results in Section 1.4.8.

1.4.1 Candidate Entities

Typically, as described in section 1.3, entity recognition has been approached by classifying individual tokens. Bunescu and Mooney [2004] considered a different approach, where candidate phrases in a document are classified according to the desired set of entity types. An advantage of using phrase classification is that it allows for phrase-based features such as the text of the candidate phrase, or its similarity with dictionary entries. However, phrase classification requires an initial set of candidate entity phrases. Considering as candidate entities all contiguous word sequences from a document would lead to a quadratic number of phrases, which would adversely affect the time complexity of the extraction algorithm. For our task, there are various heuristics that can significantly reduce the size of the candidate set; two of these are listed below:

- **H1:** In general, named entities have limited length. Therefore, one simple way of creating the set of candidate phrases is to compute the maximum length of all annotated entities in the training set, and then consider as candidates all word sequences whose length is up to this maximum length. This is also the approach followed in SRV (Freitag [1998]).
- **H2:** In the task of extracting protein names from Medline abstracts, we noticed that, like most entity names, almost all proteins in our data are base noun phrases or parts of them. Therefore, such substrings are used to determine candidate entities. To avoid missing options, we adopt a very broad definition of base noun phrase – a maximal contiguous sequence of tokens with their part-of-speech restricted to nouns, gerund verbs, past-participle verbs, adjectives, numbers and dashes. The complete set of POS tags is {*"JJ", "VBN", "VBG", "POS", "NN", "NNS", "NNP", "NNPS", "CD", "–"*} (using the Treebank notation (Marcus et al. [1993])). Also, the last word (the head) of a base noun phrase is constrained to be either a noun, or a number. Candidate extractions then consist of base NPs, together with all their contiguous subsequences headed by a noun or number.

1.4.2 Entity Features

The set of features associated with each candidate is based on the feature templates introduced in (Collins [2002]), used there for training a re-ranking algorithm on the extractions returned by a maximum-entropy tagger. Many of these features use the concept of *word type*, which allows a different form of token generalization than POS tags. The *short type* of a word is created by replacing any maximal contiguous sequences of capital letters with 'A', of lower-case letters with 'a', and of digits with '0'. For example, the word *TGF-1* would be mapped to type *A-0*.
Consequently, each token position i in a candidate extraction provides three types of information: the word itself w_i, its POS tag t_i, and its short type s_i. The full set of features types is listed in Table 1.1, where we consider a generic candidate extraction as a sequence of $n+1$ words $w_0 w_1 ... w_n$.
Each feature template instantiates numerous features. For example, the candidate extraction "HDAC1 enzyme" has the head word *HD=enzyme*, the short type *ST=A0_a*, the prefixes *PF=A0* and *PF=A0_a*, and the suffixes *SF=a* and *SF=A0_a*. All other features depend on the left or right context of the entity. Feature values that occur less than three times in the training data are filtered out.

Table 1.1 Feature Templates.

Description	Feature Template	Description	Feature Template
Text / Head	$w_0 w_1 ... w_n$ / w_n	Short Type	$s_0 s_1 ... s_n$
Bigram Left (4 bigrams)	$z_{-1} z_0$ where $z \in \{w, s\}$	Bigram Right (4 bigrams)	$z_n z_{n+1}$ where $z \in \{w, s\}$
Trigram Left (8 trigrams)	$z_{-2} z_{-1} z_0$ where $z \in \{w, s\}$	Trigram Right (8 trigrams)	$z_n z_{n+1} z_{n+2}$ where $z \in \{w, s\}$
POS Left	t_{-1}	POS Right	t_{n+1}
Prefix (n+1 prefixes)	s_0 $s_0 s_1$ ... $s_0 s_1 ... s_{n+1}$	Suffix (n+1 suffixes)	s_n $s_{n-1} s_n$ ... $s_0 s_1 ... s_{n+1}$

1.4.3 The RMN Framework for Entity Recognition

Given a collection of documents D, we associate with each document $d \in D$ a set of candidate entities $d.E$, in our case a restricted set of token sequences from the document as given by **H2** (Section 1.4.1). Each entity $e \in d.E$ is characterized by a predefined set of boolean attributes $e.F$ (Section 1.4.2), the same for all candidate entities. One particular attribute is $e.label$ which is set to 1 if e is considered a valid extraction, and 0 otherwise. In this document model, labels are the only hidden variables, and the inference procedure will try to find a most probable assignment of values to labels, given the current model parameters and the values of all other variables.
Each document is associated with a *factor graph* (Kschischang et al. [2001]), which is a bipartite graph containing two types of nodes:

- **Variable Nodes** correspond directly to the labels of all candidate entities in the document.
- **Potential Nodes** model the correlations between two or more entity attributes. For each such correlation, a *potential node* is created that is linked to all *variable nodes* involved. This is equivalent to creating a clique in the corresponding Markov random field.

The types of correlations captured by factor graphs (see Figure 1.4 for some examples) are specified by matching *clique templates* against the entire set of candidate entities $d.E$. A clique template is a procedure that finds all subsets of entities satisfying a given constraint, after which, for each entity subset, it connects through a *potential node* all the *variable nodes* corresponding to a selected set of attributes. Formally, there is a set of clique templates C, with each template $c \in C$ specified by:

1. A matching operator M_c for selecting subsets of entities, $M_c(E) \subseteq 2^E$
2. A selected set of features $S_c = \langle X_c, Y_c \rangle$, the same for all subsets of entities returned by the matching operator. X_c denotes the observed features, while Y_c refers to the hidden labels.

3. A clique potential ϕ_c which gives the compatibility of each possible configuration of values for the features in S_c, s.t. $\phi_c(s) \geq 0, \forall s \in S_c$.

Given a set E of nodes, $M_c(E)$ consists of subsets of entities whose attribute nodes S_c are to be connected in a clique. In previous applications of RMNs, the selected subsets of entities for a given template have the same size; however, some of our clique templates may match a variable number of entities. The set S_c may contain the same attribute from different entities. Usually, for each entity in a matching set, its label is included in S_c. All these will be illustrated with examples in Sections 1.4.4 and 1.4.5 where the clique templates used in our model are described in detail. Depending on the number of hidden labels Y_c selected by a clique c, we define two categories of clique templates:

- **Local Templates** are all templates $c \in C$ for which $|Y_c| = 1$. They model the correlations between an entity's observed features and its label.
- **Global Templates** are all templates $c \in C$ for which $|Y_c| > 1$. They capture influences between multiple entities from the same document.

After the factor graph model for a document d has been completed with potential nodes from all templates, the probability distribution over the random field of hidden entity labels $d.Y$ given the observed features $d.X$ is given by the Gibbs distribution:

$$P(d.Y|d.X) = \frac{1}{Z(d.X)} \prod_{c \in C} \prod_{G \in M_c(d.E)} \phi_C(G.X_c, G.Y_c) \tag{1.1}$$

where $Z(d.X)$ is the normalizing partition function:

$$Z(d.X) = \sum_{Y} \prod_{c \in C} \prod_{G \in M_c(d.E)} \phi_C(G.X_c, G.Y_c) \tag{1.2}$$

There are two problems that need to be addressed when working with RMNs:

1. **Inference:** Usually, two types of quantities are needed from an RMN model:
   - The marginal distribution for a hidden variable, or for a subset of hidden variables in the graphical model.
   - The most probable assignment of values to all hidden variables in the model.
2. **Learning:** As the structure of the RMN model is already defined by its clique templates, learning refers to finding the clique potentials that maximize the likelihood over the training data. Inference is usually performed multiple times during the learning algorithm, which means that an accurate, fast inference procedure is doubly important.

The actual algorithms used for inference and learning will be described in Sections 1.4.6 and 1.4.7 respectively.

1.4.4 Local Clique Templates

As described in the previous section, the role of local clique templates is to model correlations between an entity's observed features (see Table 1.1) and its label. For each binary feature f we introduce a local template LT_f. Given a candidate entity e, with the observed feature $e.f = 1$, the template LT_f creates a potential node linked to the variable node $e.label$. As an example, Figure 1.3 shows that part of the factor graph which is generated around the entity label for "HDAC1 enzyme", with potential nodes for the head feature (HD), prefix features (PF) and suffix features (SF). Variable nodes are shown as empty circles and potential nodes are figured as black squares. The potential ϕ_f associated with all potential nodes created by template LT_f would consist in a 1×2 table, as $e.f$ is known to be 1, and $e.label$ has cardinality 2 (assuming only one entity type is to be extracted, we need only two values for the label attribute).

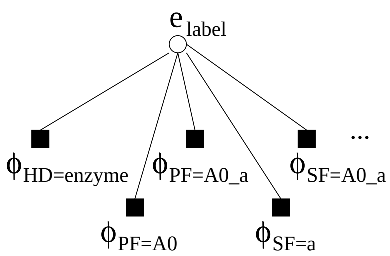


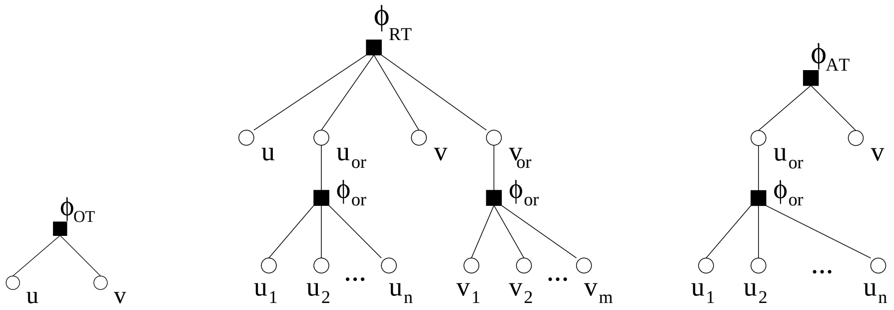
Figure 1.3 Factor Graph for local templates.

1.4.5 Global Clique Templates

Global clique templates enable us to model hypothesized influences between entities from the same document. They create potential nodes connected to the label nodes of two or more entities. In our experiments we use three global templates:

- **Overlap Template (OT):** No two entity names overlap in the text i.e if the span of one entity is $[s_1, e_1]$ and the span of another entity is $[s_2, e_2]$, and $s_1 \leq s_2$, then $e_1 < s_2$.
- **Repeat Template (RT):** If multiple entities in the same document are repetitions of the same name, their labels tend to have the same value (i.e. most of them are protein names, or most of them are not protein names). In Section 1.4.5.2 we discuss situations in which repetitions of the same protein name are not tagged as proteins, and design an approach to handle this.
- **Acronym Template (AT):** It is common convention that a protein is first introduced by its long name, immediately followed by its short-form (acronym) in parentheses.

In Figure 1.4 we show the factor graphs created by these global templates, each of which is explained in the following sections.



(a) Overlap Factor Graph (b) Repeat Factor Graph (c) Acronym Factor Graph

Figure 1.4 Factor Graphs for global templates.

1.4.5.1 The Overlap Template

The definition of a *candidate extraction* from Section 1.4.1 leads to many overlapping entities. For example, "glutathione S - transferase" is a base NP, and it generates five candidate extractions: "glutathione", "glutathione S", "glutathione S - transferase", "S - transferase", and "transferase". If "glutathione S - transferase" has label-value 1, the other four entities should all have label-value 0, because they overlap with it.
This type of constraint is enforced by the overlap template by creating a potential node for each pair of overlapping entities and connecting it to their label nodes, as shown in Figure 1.4(a). To avoid clutter, all entities in this and subsequent factor graphs stand for their corresponding labels. The potential function ϕ_{OT} is manually set so that at most one of the overlapping entities can have label-value 1, as illustrated in Table 1.2.

Table 1.2 Overlap Potential.

ϕ_{OT}	$e_1.label = 0$	$e_1.label = 1$
$e_2.label = 0$	1	1
$e_2.label = 1$	1	0

Continuing with the previous example, because "glutathione S" and "S - transferase" are two overlapping entities, the factor graph model will contain an overlap potential node connected to the label nodes of these two entities.

1.4.5.2 The Repeat Template

We could specify the potential for the repeat template in a 2×2 table, this time leaving the table entries to be learned, given that assigning the same label to repetitions is not a hard constraint. However we can do better by noting that the vast majority of cases where a repeated protein name is not also tagged as a protein happens when it is part of a larger phrase that *is* tagged. For example, "HDAC1 enzyme" is a protein name, therefore "HDAC1" is not tagged in this phrase, even though it may have been tagged previously in the abstract where it was not followed by "enzyme". We need a potential that allows two entities with the same text to have different labels if the entity with label-value 0 is inside another entity with label-value 1. But a candidate entity may be inside more than one "including" entity, and the number of including entities may vary from one candidate extraction to another. Using the example from Section 1.4.5.1, the candidate entity "glutathione" is included in two other entities: "glutathione S" and "glutathione S - transferase".
In order to instantiate potentials over a variable number of label nodes, we introduce a *logical OR clique template* that matches a variable number of entities. When this template matches a subset of entities $e_1, e_2, ..., e_n$, it will create an auxiliary OR entity e_{OR}, with a single attribute $e_{OR}.label$. The potential function ϕ_{OR} is manually set so that it assigns a non-zero potential only when $e_{OR}.label = e_1.label \vee e_2.label \vee ... \vee e_n.label$. The potential nodes are only created as needed, e.g. when the auxiliary OR entity is required by repeat and acronym clique templates.
Figure 1.4(b) shows the factor graph for a sample instantiation of the repeat template using the OR template. Here, u and v represent two same-text entities, u_1, u_2, ... u_n are all entities that include u, and v_1, v_2, ..., v_m are entities that include v. The potential function ϕ_{RT} can either be manually preset to prohibit unlikely label configurations, or it can be learned to represent an appropriate soft constraint. In our experiments, it was learned since this gave slightly better performance.
Following the previous example, suppose that the phrase "glutathione" occurs inside two base NPs in the same document, "glutathione S - transferase" and "glutathione antioxidant system". Then the first occurrence of "glutathione" will be associated with the entity u, and correspondingly its including entities will be $u_1 =$ "glutathione S" and $u_2 =$ "glutathione S - transferase". Similarly, the second occurrence of "glutathione" will be associated with the entity v, with the corresponding including entities $v_1 =$ "glutathione antioxidant" and $v_2 =$ "glutathione antioxidant system".

1.4.5.3 The Acronym Template

One approach to the acronym template would be to use an extant algorithm for identifying acronyms and their long forms in a document, and then define a potential function that would favor label configurations in which both the acronym and its definition have the same label. One such algorithm is described by Schwartz and Hearst [2003], achieving a precision of 96% at a recall rate of 82%. However, because this algorithm would miss a significant number of acronyms, we have decided to implement a softer version as follows: detect all situations in which a single word is enclosed between parentheses, such that the word length is at least 2 and it begins with a letter. Let v denote the corresponding entity. Let u_1, u_2, ..., u_n be all entities that end exactly before the open parenthesis. If this is a situation in which v is an acronym, then one of the entities u_i is its corresponding long form. Consequently, we use a logical OR template to introduce the auxiliary entity u_{OR}, and connect it to v's node label through an acronym potential ϕ_{AT}, as illustrated in Figure 1.4(c). For example, consider the phrase "the antioxidant superoxide dismutase - 1 (SOD1)". "SOD1" satisfies our criteria for acronyms, thus it will be associated with the entity v in Figure 1.4(c). The candidate long forms are u_1 = "antioxidant superoxide dismutase - 1", u_2 = "superoxide dismutase - 1", and u_3 = "dismutase - 1".

1.4.6 Inference in Factor Graphs

In our setting, given the clique potentials, the inference step for the factor graph associated with a document involves computing the most probable assignment of values to the hidden labels of all candidate entities:

$$d.Y^* = arg \max_{d.Y} P(d.Y|d.X) \tag{1.3}$$

where $P(d.Y|d.X)$ is defined as in Equation 1.1. A brute-force approach is excluded, since the number of possible label configurations is exponential in the number of candidate entities. The sum-product algorithm (Kschischang et al. [2001]) is a message-passing algorithm that can be used for computing the marginal distribution over the label variables in factor graphs without cycles, and with a minor change (replacing the sum operator used for marginalization with a max operator) it can also be used for deriving the most probable label assignment. In our case, in order to get an acyclic graph, we would have to use local templates only. However, it has been observed that the algorithm often converges in general factor graphs, and when it converges, it gives a good approximation to the correct marginals. The algorithm works by altering the belief at each label node by repeatedly passing messages between the node and all potential nodes connected to it (Kschischang et al. [2001]).

1.4.7 Learning Potentials in Factor Graphs

Following a maximum likelihood estimation, we shall use the log-linear representation of potentials:

$$\phi_C(G.X_c, G.Y_c) = exp\{\mathbf{w_c}\mathbf{f_c}(G.X_c, G.Y_c)\} \tag{1.4}$$

Let $\mathbf{w}$ be the concatenated vector of all potential parameters $\mathbf{w_c}$. One approach to finding the maximum-likelihood solution for $\mathbf{w}$ is to use a gradient-based method, which requires computing the gradient of the log-likelihood with respect to potential parameters $\mathbf{w_c}$. It can be shown that this gradient is equal with the difference between the empirical counts of $\mathbf{f_c}$ and their expectation under the current set of parameters $\mathbf{w}$.

$$\nabla L(w, D) = \sum_{d \in D} f_c(d.X, d.Y) - \sum_{d \in D} \sum_{d.Y'} f_c(d.X, d.Y') P_w(d.Y'|d.X) \tag{1.5}$$

The expectation in the second term is expensive to compute, since it requires summing over all possible configurations of candidate entity labels from a given document. To circumvent this complexity, we used the voted perceptron approach (Freund and Schapire [1999]), which can be seen as approximating the full expectation of $\mathbf{f_c}$ with the $\mathbf{f_c}$ counts for the most likely labeling under the current parameters $\mathbf{w}$.

$$\nabla L(w, D) \approx \sum_{d \in D} f_c(d.X, d.Y) - \sum_{d \in D} f_c(d.X, d.Y_w) \tag{1.6}$$

The Voted Perceptron algorithm is detailed in Table 1.3. At each step i in the

Table 1.3 The Voted Perceptron Algorithm.

Input: a set of documents D, number of epochs T, learning rate η.
set parameters $w_0 = 0$, counter $i = 0$
for $t = 1...T$
for every document $d \in D$
$d.Y_i = arg\max_{d.Y'} P_{w_i}(d.Y'|d.X)$
$w_{i+1} = w_i + \eta * [f(d.X, d.Y) - f(d.X, d.Y_i)]$
$i = i + 1$
Output: $w = \frac{1}{T\|D\|} \sum_i w_i$

algorithm, inference is performed using the current parameters w_i, which results in the most likely labeling $d.Y_i$. The parameters are then updated based on the difference between the features counts computed on the ideal labeling $d.Y$ and those computed on the current most likely labeling $d.Y_i$. The final set of parameters is the average taken over the parameters at all steps i in the algorithm. In all our experiments, the perceptron was run for 50 epochs, with a learning rate set at 0.01.

1.4.8 Experimental Results

We have tested the RMN approach on two datasets that have been hand-tagged for human protein names. The first dataset is Yapex[1] which consists of 200 Medline abstracts. The second dataset is Aimed[2], which consists of 225 Medline abstracts we previously annotated for evaluating systems that extract both human proteins

1. URL:www.sics.se/humle/projects/prothalt/
2. URL:ftp.cs.utexas.edu/mooney/bio-data/

and their interactions (Bunescu et al. [2005]).
We compared the performance of three systems:

- **LT-RMN** is the RMN approach using local templates and the overlap template.
- **GLT-RMN** is the full RMN approach, using all local and global templates.
- **CRF**, which uses a CRF for labeling token sequences. We used the CRF implementation from (McCallum [2002]) with the set of tags and features employed by the Maximum-Entropy tagger described in (Bunescu et al. [2005]).

All Medline abstracts were tokenized and then POS tagged using the (Brill [1995]) tagger. Each extracted protein name in the test data was compared to the human-tagged data, with the positions taken into account. Two extractions are considered a match if they consist of the same character sequence in the same position in the text.
Results are shown in Table 1.4, which presents the standard IE metrics of average *precision* (percentage of extracted names that are correct), *recall* (percentage of correct names that are extracted), and *F-measure* (harmonic mean of precision and recall) using 10-fold cross validation.

Table 1.4 Information-Extraction Performance on Two Human Protein Corpora.

Yapex			
Method	Precision	Recall	F-m
LT-RMN	70.79	53.81	61.14
GLT-RMN	69.71	65.76	67.68
CRF	72.45	58.64	64.81

Aimed			
Method	Precision	Recall	F-m
LT-RMN	81.33	72.79	76.82
GLT-RMN	82.79	80.04	81.39
CRF	85.37	75.90	80.36

In terms of F-measure, the use of global templates for modeling influences between possible entities from the same document significantly improves extraction performance over the local approach (a one-tailed paired t-test for statistical significance results in a p value less than 0.01 on both datasets). There is also a small improvement over CRFs, with the results being statistically significant only for the Yapex dataset, corresponding to a p value of 0.02. As expected, GLT-RMN gave a consistently higher recall – additional protein names were extracted as a result of linking them to repetitions with more informative contexts.
We hypothesize that further improvements to the LT-RMN approach and a better inference algorithm would push the GLT-RMN performance even higher. In (Bunescu [2004]), based on a version of the junction tree algorithm that exploits the sparsity of the overlap potential, we show that exact inference for the LT-RMN case can be performed efficiently, with time complexity linear in terms of the number of candidate entities. In the same work, it is shown that if the candidate entities are given by the weak (but complete) heuristic H1, the new LT-RMN approach can be used for returning all text positions that are unlikely to belong to a named entity. This provides a general method for reducing the number of candidate extractions,

which can replace the domain-dependent heuristic H2. The main drawback of this heuristic is that sometimes it may miss true entity names - its coverage is 95.6% on Yapex and 97.1% on Aimed. As an example, H2 assumes that a candidate entity cannot contain parentheses, however the Yapex corpus contains a few entity names like "V (1a) receptor", or "interleukin 10 (IL-10) receptor", which violate this assumption. Instead, the local phrase model can be used to learn patterns like "allow a close parenthesis in an entity name if it is followed by the word *receptor*".
For the global model GLT-RMN, the inference procedure can be improved by using a tree-based message propagation schedule, also known as tree reparameterization (TRP) (Wainwright et al. [2001]). TRP has the advantage that if often converges in cases where the sum-product algorithm fails, requiring a considerably shorter time for convergence.

1.5 Future Research on SRL for NLP

There are a variety of promising directions for future research in applying SRL to NLP. With respect to information extraction, in addition to identifying entities, an important problem is extracting specific types of relations between entities. For example, in newspaper text, one can identify that an organization is located in a particular city or that a person is affiliated with a specific organization (Zelenko et al. [2003]); in biomedical text, one can identify that a protein interacts with another protein or that a protein is located in a particular part of the cell (Bunescu and Mooney [2005], Craven and Kumlien [1999]). SRL methods may be usefully applied to such problems since they require identifying relations between phrases that occur in different parts of a sentence or paragraph.
The complete task of natural language understanding incorporates a wide variety of interacting subtasks such as, speech recognition, morphology, part-of-speech tagging, phrase chunking, syntactic parsing, word-sense disambiguation, semantic interpretation, anaphora (e.g. pronoun) resolution, and discourse processing. Each of these tasks requires disambiguating between numerous possibilities and resolving each of these ambiguities interacts in complex ways with many of the others. For example, when understanding the passage: "At the zoo, several men were showing a group of students various types of flying animals. Suddenly, one of the students hit the man with a bat." one must first use the context in the previous sentence to resolve the meaning of the word "bat" before being able to properly attach the misleading prepositional phrase "with a bat" to the man (NP) rather than to the hitting (VP). SRL methods hold the promise of being able to integrate decisions at all levels of syntactic, semantic, and pragmatic processing in order to correctly interpret natural language. Several recent projects have taken the first steps in this direction. For example, (Sutton et al. [2004]) present a dynamic version of a CRF that integrates part-of-speech tagging and noun-phrase chunking into one coherent process. (Roth and Yih [2004]) present an information-extraction approach based on linear-programming that integrates recognition of

entities with the identification of relations between these entities. The ability of SRL techniques to integrate uncertain evidence from many interacting problems in order to collectively determine a globally coherent solution to all of them could help develop a complete, robust NLP system. However, such a system would create massive collective inference problems and would require efficient SRL methods that could scale to very large networks.

1.6 Conclusions

The area of natural language processing includes many problems that lend themselves to SRL methods. Most existing statistical methods in NLP such as HMMs, sequence CRFs and PCFGs are actually restrictive forms of SRL. More general SRL techniques have advantages over these existing methods and hold the promise of improving results on a number of difficult NLP problems. In this paper, we have reviewed our research on applying SRL techniques to information extraction. By using RMNs to capture dependencies between distinct candidate extractions in a document, we achieved improved results on identifying names of proteins in biomedical abstracts compared to a traditional CRF. By using the ability of SRL to integrate disparate sources of evidence to perform collective inference over complex relational data, robust NLP systems that accurately resolve many interacting ambiguities can hopefully be developed.

Acknowledgements

This research was partially supported by the National Science Foundation under grants IIS-0325116 and IRI-9704943.

References

James F. Allen. *Natural Language Understanding*. Benjamin/Cummings, Menlo Park, CA, 1987.

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

Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun Kumar Ramani, and Yuk Wah Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine (special issue on Summarization and Information Extraction from Medical Documents)*, 33(2):139–155, 2005.

Razvan C. Bunescu. Learning for collective information extraction. Technical Report TR-05-02, Department of Computer Sciences, University of Texas at Austin, 2004.

Razvan C. Bunescu and Raymond J. Mooney. Collective information extraction with relational Markov networks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 439–446, Barcelona, Spain, July 2004.

Razvan C. Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In *Advances in Neural Information Processing Systems*, Vancouver, BC, 2005. To appear.

Claire Cardie. Empirical methods in information extraction. *AI Magazine*, 18(4): 65–79, 1997.

Kenneth W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143, Austin, TX, 1988. Association for Computational Linguistics.

Michael 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-2002)*, pages 489–496, Philadelphia, PA, 2002.

M. Craven and 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 (ISMB-1999)*, pages 77–86, Heidelberg, Germany, 1999.

Jenny R. Finkel, Trond Grenager, and Christopher D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, pages 363–370, Ann Arbor, MI, 2005.

Dayne Freitag. Information extraction from HTML: Application of a general learning approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 517–523, Madison, WI, July 1998. AAAI Press / The MIT Press.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296, 1999.

J. Hirschberg. Every time I fire a linguist, my performance goes up, and other myths of the statistical natural language processing revolution. Invited talk, Fifteenth National Conference on Artificial Intelligence (AAAI-98), 1998.

Frederick Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, 1976.

Frederick Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, MA, 1998.

F. R. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of 18th International Conference on Machine Learning (ICML-2001)*, pages 282–289, Williamstown, MA, 2001.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA, 1999.

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

Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

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

A. K. Ramani, R. C. Bunescu, R. J. Mooney, and E. M. Marcotte. Consolidating the set of know human protein-protein interactions in preparation for large-scale mapping of the human interactome. *Genome Biology*, 6(5):r40, 2005.

Lance A. Ramshaw and Mitchell P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora*, 1995.

D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 1–8, Boston, MA, 2004.

Roger C. Schank and Christopher K. Riesbeck. *Inside Computer Understanding: Five Programs plus Miniatures*. Lawrence Erlbaum and Associates, Hillsdale, NJ, 1981.

Ariel S. Schwartz and Marti A. Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of the 8th Pacific Symposium on Biocomputing*, pages 451–462, Lihue, HI, January 2003.

Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*, Banff, Canada, 2004.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of 21st International Conference on Machine Learning (ICML-2004)*, pages 783–790, Banff, Canada, July 2004.

Benjamin Taskar, Pieter Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of 18th Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, pages 485–492, Edmonton, Canada, 2002.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree-based reparameterization framework for approximate estimation on graphs with cycles. In *Advances in Neural Information Processing Systems*, 2001.

Terry Winograd. *Understanding Natural Language*. Academic Press, Orlando, FL, 1972.

William A. Woods. Lunar rocks in natural English: Explorations in natural language question answering. In Antonio Zampoli, editor, *Linguistic Structures Processing*. Elsevier North-Holland, New York, 1977.

D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.