Learning for Collective Information Extraction

Razvan C. Bunescu
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
razvan@cs.utexas.edu

Doctoral Dissertation Proposal

Supervising Professor: Raymond J. Mooney

Abstract

An Information Extraction (IE) system analyses a set of documents with the aim of identifying certain types of entities and relations between them. Most IE systems treat separate potential extractions as independent. However, in many cases, considering influences between different candidate extractions could improve overall accuracy. For example, phrase repetitions inside a document are usually associated with the same entity type, the same being true for acronyms and their corresponding long form. One of our goals in this thesis is to show how these and potentially other types of correlations can be captured by a particular type of undirected probabilistic graphical model. Inference and learning using this graphical model allows for "collective information extraction" in a way that exploits the mutual influence between possible extractions. Preliminary experiments on learning to extract named entities from biomedical and newspaper text demonstrate the advantages of our approach.

The benefit of doing collective classification comes however at a cost: in the general case, exact inference in the resulting graphical model has an exponential time complexity. The standard solution, which is also the one that we used in our initial work, is to resort to approximate inference. In this proposal we show that by considering only a selected subset of mutual influences between candidate extractions, exact inference can be done in linear time. Consequently, a short term goal is to run comparative experiments that would help us choose between the two approaches: exact inference with a restricted subset of mutual influences or approximate inference with the full set of influences.

The set of issues that we intend to investigate in future work is two fold. One direction refers to applying the already developed framework to other natural language tasks that may benefit from the same types of influences, such as word sense disambiguation and part-of-speech tagging. Another direction concerns the design of a sufficiently general framework that would allow a seamless integration of cues from a variety of knowledge sources. We contemplate using generic sources such as external dictionaries, or web statistics on discriminative textual patterns. We also intend to alleviate the modeling problems due to the intrinsic local nature of entity features by exploiting syntactic information. All these generic features will be input to a feature selection algorithm, so that in the end we obtain a model which is both compact and accurate.

Contents

1 Introduction

Information Extraction (IE) is an important task in natural language processing, with many practical applications. It involves analyzing text documents, identifying particular types of entities, relations or events, and populating database slots with information about them. A basic component of an IE system is that of named entity recognition - the task of locating references to specific types of items in natural-language text. Since IE systems are difficult and time-consuming to construct, most recent research has focused on empirical techniques that automatically construct information extractors by training on supervised corpora (Cardie, 1997; Califf, 1999). Traditionally, IE systems have been trained to recognize names of people, organizations and locations (MUC (Grishman, 1995), CoNLL (Tjong Kim Sang & De Meulder, 2003)). Recently, substantial resources have been allocated for automatically extracting information from biomedical corpora, which has naturally led to the need of locating biologically relevant entity types, such as genes, proteins, or diseases. The wide variety of names used in the biomedical literature, coupled with their lack of formal structure, have made the IE problem especially difficult. This has further motivated the search for methods which are able to efficiently use any type of task-relevant knowledge. One particular type of knowledge which is especially useful for recognizing biological entities refers to 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 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: Medline abstract with all protein names emphasized.

It is not always the case that repeated phrases have the same label. Figure 2 shows an example, where the first occurrence of 'eNOS' is a protein name, while its second occurrence is not a protein name by itself, because it is included in another protein name 'eNOS interaction protein'. Constraining repeated words like 'eNOS' to have the same label (i.e. either **I**nside or **O**utside a protein name) does not solve the problem either, as shown in Figure 2, where both tokens 'nitric' and 'oxide' are first tagged as **O**utside, and then **I**nside a protein name. In Section 3.5.2 we show how to capture the correlations between the labels of repeated phrases so that all the exceptions above are taken into account.

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 (see Figure 3). Therefore, correlations between the labels of repeated phrases, or between acronyms and

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 used the yeast two - hybrid system and 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 2: Medline abstract with all protein names emphasized.

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.

The 5′ upstream region (-448 / -443) of the human **dipeptidyl peptidase IV** gene promoter containing a consensus E - box (CACGTG) was shown to bind upstream stimulatory factor using nuclear extracts from mouse (3T3) fibroblasts and the human intestinal and hepatic epithelial cell lines Caco - 2 and HepG2 .

Figure 3: Medline abstract with all protein names emphasized.

In this proposal, we describe how this type of document-level knowledge can be captured using Relational Markov Networks (RMNs) (Taskar, Abbeel, & Koller, 2002), a version of undirected graphical models which have already been successfully used to improve the classification of hyper-linked web pages. While other types of graphical models, such as Conditional Random Fields (CRFs) (Lafferty, McCallum, & Pereira, 2001), have modeled the entity recognition task as one of token classification, we take the different approach where candidate phrases in a document are classified according to the desired set of entity types. We then show how this phrase classification approach facilitates the modeling of correlations among labels of candidate entities, with the additional strength of phrase based features such as the actual text of the candidate entity, its capitalization pattern, or similarity with dictionary entries. Experimental results show that by factoring in global label correlations, the performance of the phrase classification approach is significantly improved.

In a typical application of CRFs, influences between the labels of consecutive tokens are the only correlations considered. This leads to a sequence labeling scenario, in which inference can be done efficiently using dynamic programming algorithms. Compared with CRFs, the increased representational power of RMNs comes at a cost in the time complexity of the inference algorithms. In our initial work, we resorted to approximate inference, based on an algorithm which has already exhibited competitive performance in other applications (Murphy, Weiss, & Jordan, 1999). However, in subsequent work we have discovered that exact inference for the phrase classification approach can be done efficiently, if no correlations between different candidate entities are to be considered. Moreover, by adding a carefully selected subset of document-level correlations, the same exact inference algorithm can be updated so that its running time remains linear in the number of candidate entities. We present how to select the correlations that are to be included in the model, and prove the linear complexity of the inference algorithm when run on the resulting structure.

As short-term goals, we intend to compare the exact inference / limited set of correlations approach with the original approach based on approximate inference / full set of correlations. Another direction that might lead to improved results is that of using a generalized version of the belief propagation algorithm, where messages are passed between sets of nodes, at additional computational cost (Yedidia, Freeman, & Weiss, 2000), or using alternative approximate-inference methods.

Long-term goals include:

- Applying the same approach to other natural language tasks that may benefit from document-level correlations. Two examples are word sense disambiguation (WSD) and part-of-speech (POS) tagging. In WSD, the one sense per discourse hypothesis has been previously used by Yarowsky in (Yarowsky, 1995). The RMN framework however is able to incorporate this type of knowledge in a more probabilistic, sound manner. As for the task of POS tagging, while the benefit of using correlations between tags of repeated words is debatable if the tagger is trained and tested on documents from the same corpora, given its already competitive performance, it has nevertheless the potential of reducing the number of tagging errors on texts from different corpora.

- Using features based on similarities with existing dictionary entries. Such features can be incorporated in our phrase classification approach in a natural, straightforward manner.

- The ”web as a corpus” is still an under-utilized idea. In this context, textual patterns that disambiguate the type of a candidate entity can be provided to a web search engine, so that statistics derived from the number of returned hits may be used in order to increase the IE system’s performance.

- Syntactic information has already been proved to increase the accuracy on the task of relation extraction. Besides designing an IE system for extracting relations specific to biomedical entities, we intend to leverage entity recognition through the use of features derived from syntactic parses. Some of these features have the benefit of encoding long-range dependencies which cannot be captured from a flat representation of sentences.

- The previous three goals can be seen as part of the effort to design a general framework that would allow the use of information from various knowledge sources in order to increase the final IE system’s performance. We intend to increase the robustness of this approach through the use of an efficient feature selection algorithm.

2 Background and Related Work

The task of automatically constructing information extractors has received a lot of attention in the past decade, and as such we observe a high diversity in the proposed approaches and the learning algorithms used therein. Nevertheless, a careful analysis reveals that most of these systems can be classified into two basic types of approaches:

- **Token Classification**: Word tokens in a document are sequentially classified as being inside or outside of a given named entity. Named entities are extracted by doing token classification and then assembling maximally contiguous sequences of inside tokens.

- **Phrase Classification**: Candidate phrases from a document are classified as to whether they are instances of some entity types or not. This can be done by either learning a multi-class classifier, in which case the number of classes is equal with the number of entity types plus one (for non-entity phrases), or by separately learning sets of extraction patterns, one set of patterns for each of the entity types.

2.1 Phrase Classification Approaches

Relational learning has been one of the learning paradigms used in some of the early IE systems, such as Rapier (Califf & Mooney, 1999) and SRV (Freitag, 1998). Both systems belong to the phrase classification approach.

2.1.1 Rapier

In RAPIER, the IE task is defined in terms of filling the slots contained in a template. A template specifies a particular type of event, such as joint ventures, corporate acquisitions, or job offerings. For example, a job offering template contains slots for title, salary, area of expertise, OS platform required, or job location. The training data consists of filled templates, one template per document. During testing, the IE system fills the template slots with data extracted from the document.

For each template slot, a set of rules is learned in a bottom-up fashion, with each rule composed of patterns that can make use of limited syntactic information. More exactly, the extraction rules consist of three parts:

1. A pre-filler pattern that matches text immediately preceding the slot filler,
2. A pattern that matches the actual field, and
3. A post-filler pattern that matches the text immediately following the slot filler.

Each pattern is a sequence of pattern elements of one of two types: *pattern items* and *pattern lists*. A pattern item matches exactly one word that satisfies its constraints. A pattern list has a maximum length N and matches 0 to N words, each satisfying a set of constraints. Besides constraining on words and their part-of-speech tags, Rapier can also incorporate semantic class information, such as that provided by WordNet (Miller, 1991). Consequently, each constraint is represented as a disjunctive list of one or more words, tags or WordNet synsets.

During testing, phrases are extracted by matching them against the set of rules learned for each slot. For the template filling task, extracted phrases which are duplicated are ignored, however the system can be easily modified to work in a named entity scenario, such that the output contains all extracted phrases, duplicates included. Because each pattern is designed to match phrases, we can view Rapier as belonging to the generic class of phrase classification approaches.

2.1.2 SRV

SRV (Freitag, 1998) too is based on a relational learning procedure. Like FOIL (Quinlan, 1990), it proceeds in a top-down fashion, starting with the entire set of examples - all negative examples and any positive examples not covered by already induced rules. At each step it greedily adds predicates, trying to cover as many positive, and as few negative examples as possible. There is a set of predefined predicate templates including tests on the length of the candidate entity or tests on features of tokens inside the candidate phrases. Token features are predefined too and come in two categories:

- *simple* features such as the word, its capitalization pattern, binary features testing whether the token is a punctuation sign, or a number.
- two *relational* features - the previous and the next tokens.

As with any phrase classification approach, SRV needs to address the issue of searching through a typically huge negative examples set. The authors do this by handling negative examples implicitly, on a token-by-token basis – examples are indexed based on the tokens they contain. Because a token is generally shared by many candidate phrases, this leads to a more tractable search method.

2.2 Token Classification Approaches

2.2.1 Hidden Markov Models

Another class of approaches to learning IE systems is based on Hidden Markov Models (HMMs) (Rabiner, 1989). HMMs have been successfully used for speech recognition before becoming a model of choice for other natural language tasks such as POS tagging or named entity recognition. An HMM can be defined as the stochastic version of a finite state automaton. Thus, there is a set of states (hidden), with transitions between them. Given a state, there is a probability distribution over all possible transitions from that state. Symbols can be generated from any state, one symbol at a time, based on a symbol emission distribution. In a typical application of HMMs, a sequence of symbols is given, together with an HMM that is assumed to have produced it. The generative process by which the HMM produces a string of symbols starts by choosing a distinguished state (referred to as a starting state), then transitioning to another state according the the corresponding transition probability. This process of transitioning from one state to another continues until it reaches another distinguished state (referred to as the final state). Each time a transition is made from a state, a symbol is generated according to that state's symbol emission probability distribution. Graphically, an unrolled HMM can be represented as a directed graph, as in Figure 4. In this and all subsequent figures, the X symbols are used to denote observations, while Y symbols refer to hidden variables (states or labels).

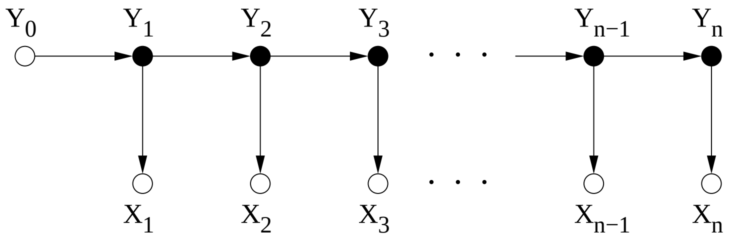

Figure 4: Unrolling an HMM as a directed graphical model

One of the questions that an HMM inference algorithm is usually required to answer is what is the sequence of states that is most likely to have generated a given sequence of symbols. For example, in the case of POS tagging, each state corresponds to a POS tag, whereas symbols correspond to words. Given a particular sentence, the POS tagging is defined as the most likely sequence of states that generated the sentence.

HMMs are particularly attractive as they have a solid mathematical foundation, and the associated inference problem can be solved in time linear with the number of observed symbols using dynamic programming (the Viterbi algorithm). During learning, if the data is fully observable (e.g. labeled training data), the HMM parameters are simply set to their maximum likelihood estimates. If the data is only partially observable i.e. the states are hidden, the Baum-Welch algorithm, an instantiation of the more general Expectation Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977), can be used to find a set of parameters such that the likelihood function is locally optimized.

IE systems based on HMMs belong naturally to the category of token classification approaches. The most likely path through the Markov model leads to a tagging of the input symbols, and consequently entities are extracted by assembling maximal contiguous sequences of words which are tagged with the same entity tag.

There exist numerous IE systems based on HMMs, and with them a whole diversity of augmentations to the basic model was introduced in order to better address various aspects of the task, such as the need for adequate representational power, or how to deal with sparsity due to insufficient training data.

Nymble. Nymble (Bikel, Schwartz, & Weischedel, 1999) is one of the earliest learning systems for named entity recognition based on HMMs. It consists of an ergodic model with one state for each entity type, together with an additional state for tokens outside any entity. Inside each name-class state, words are generated based on a statistical bigram language model. The generation of name-classes (states) and words proceeds in three steps, which are repeated until the entire word sequence is observed.

1. Select a name-class, conditioning on the previous name-class and the previous word.

2. Generate the first word from inside that name-class, by conditioning on the current and previous name-classes.

3. Generate subsequent words inside the current name-class, where each word is conditioned on its immediate predecessor.

In this model, words are generally assimilated with ordered pairs of words and word features $\langle w, f \rangle$, where features belong to a predefined set of features, similar with those used in the SRV system. This further exacerbates the problem of insufficient training data for estimating the model parameters. Consequently, the authors rely on a multi-level back-off scheme, with weights for each level of back-off set based on an empirical formula.

HMMs and Shrinkage. A different approach is proposed in (Freitag & McCallum, 1999), where a separate HMM is created for each of the extraction fields. The states in each HMMs are either background or target states. Prefix and suffix states are distinguished from other background states in order to account for distributional peculiarities in the case of tokens occurring before or after the target field. Similarly, because certain tokens tend to occur at the beginning or end of the fragment, the target state is expanded into an array of parallel paths of varying length. The problem of data sparsity is alleviated through the use of ”shrinkage”, a statistical technique which combines parameter estimates from data-sparse states of a complex model with estimates from data-rich states of a simpler model. The method relies on a hierarchy that represents expected similarity between parameter estimates, with the estimates of the complex model at the leaves. In the case of shrinkage for HMM, subsets of states having similar word emission distributions are connected to a common parent. Internal nodes in turn can share a common parent, thus encoding weaker similarities between the corresponding groups of states. Word emission probabilities associated with states high in the hierarchy become simpler than those for states below, with the top of each hierarchy corresponding to the uniform distribution. The ”shrinkage-based ” parameter estimate is defined as a linear interpolation of the estimates in all distributions from the leaf to the root. The corresponding mixture weights are optimized by running EM on a held out dataset.

HMMs and Structure Learning. The two recently discussed HMM-based systems start with a predefined model structure, and learning is used only in estimating the model parameters. For tasks in which the entities to be extracted are densely represented inside a document, as is the case with headers and research paper references, a single HMM containing states for all entity types may be more appropriate. Variability in the relative ordering of the fields can be captured in the model by allowing the same field to be represented by more than one state. Learning the structure of such a model is the focus of the approach described in (Seymore, McCallum, & Rosenfeld, 1999).

2.2.2 Discriminative Models

We have already distinguished between IE approaches based on *token classification* and approaches based on *phrase classification*. Another useful dichotomy, orthogonal to the previous one, is that of *generative* vs. *discriminative* models. All HMM models reviewed here are generative in the sense that they try to model both the observation and hidden state sequences. However, in most application of HMMs, the observations are given, the task being that of ”decoding” the hidden state sequence. Therefore, a major drawback of generative models is that modeling effort is spent on observations, instead of being focused entirely on describing the state sequence. The attempt to model the observations while keeping the inference tractable has led to the *output independence assumption*, which stipulates that the current observation, given the current state, is independent of previous observations. Usually, in text applications, observations correspond to words, and consequently the output independence assumption is not fair enough. The mismatch between model assumptions and data becomes even more pronounced if overlapping features, such as word capitalization and part-of-speech, are added as observations. Another inadequacy (McCallum, Freitag, & Pereira, 2000) is due to the way parameters are estimated. In an HMM, parameters are set to maximize the likelihood of the observation sequence, while the task is that of predicting the state sequence given the observations. All these mismatches and limitations are eliminated in discriminative approaches, in which the conditional probability of state sequences given the observations lies at the core at the model.

2.2.3 Maximum Entropy Models

The Maximum Entropy (MaxEnt) (Berger, Della Pietra, & Della Pietra, 1996) principle has been widely used to create discriminative probabilistic models for natural language tasks. The classification problem is viewed in terms of a random process that produces an output value y from a finite set Y, based on the contextual information x, a member of a finite set X. In a token classification scenario, this means associating a tag y to each text token, whereas the context x is derived from the text centered at the current token position. In maximum entropy modeling we are looking for a probability distribution $p(y|x)$ that satisfies a set of constraints $C_i \in C$ derived from a collection of user specified features $f_i(x, y) \in F$. Each feature is expressed as a binary function based on the context x at the current token position and its proposed classification y. For example, a useful feature in tagging for named entity recognition is the capitalization of the token to be classified, and it can be expressed as follows:

$$f_i(x,y) = \begin{cases} 1 & \text{if current token is capitalized \& } y = Inside, \\ 0 & \text{otherwise}. \end{cases}$$

The constraint C_i associated with a feature function f_i is expressed simply by imposing that the expected value of f_i under the target distribution $p(y|x)$ be the same as the expected value of f_i under the empirical distribution $\tilde{p}(x, y)$ (derived from the training data):

$$C_i \to \sum_{x,y} \tilde{p}(x,y) f_i(x,y) = \sum_{x,y} \tilde{p}(x) p(y|x) f_i(x,y)$$

Out of a potentially infinite number of probability distributions $p(y|x)$ satisfying a particular set of constraints, the maximum entropy principle dictates that we select the most ”uniform” distribution, where a formal measure for the ”uniformity” of a distribution is given by the information theoretic notion of conditional entropy (Cover & Thomas, 1991):

$$H(Y|X) = -\sum_{x,y} \tilde{p}(x)p(y|x)\log p(y|x)$$

Based on the concept of duality from constrained optimization, it can be shown that the distribution $p(y|x)$ satisfying the constraints C_i, and which also minimizes the conditional entropy $H(Y|X)$, is a member of the exponential family:

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

where $Z(x) = \sum_y exp\left(\sum_i \lambda_i f_i(x,y)\right)$ is a normalizing constant. An additional compelling justification for the maximum entropy principle is that the resulting distribution is also the model which, among all log-linear models of the above form, maximizes the likelihood of the parameters given the training sample.

In (Ratnaparkhi, 1996), the authors describe a maximum entropy approach to part-of-speech tagging in which they introduce a feature template $f\langle a,b\rangle$ which relates the tags of two consecutive tokens:

$$f\langle a,b\rangle(x_t, y_t, y_{t-1}) = \begin{cases} 1 & \text{if } y_{t-1} = a \text{ \& } y_t = b, \\ 0 & \text{otherwise}. \end{cases}$$

They also define a similar feature template relating the tags of three consecutive tokens. Computing the highest probability label for each token, from left to right, does not necessarily lead to the most likely sequence of tags. To alleviate this, the authors use a beam search procedure, in which they consider tokens from left to right, keeping at each position the five sequences of tags concentrating the most probability mass. A more rigorous approach, which was later used in maximum entropy models for named entity recognition (Chieu & Ng, 2003), is to use a Viterbi-like algorithm for decoding, which guarantees finding the most likely labeling of the entire sequence of words.

Maximum Entropy Markov Models. This new type of features, relating tags in consecutive positions, suggests a class of maximum entropy models in which binary features may include a test on the class of the previous token, besides conditioning on the observed input context and the mandatory test on the class of the current token. Each such feature is uniquely identifiable by a condition g on the observed input x_t and the possible instantiations a and b for the current and previous tags, y_t and y_{t-1}, as follows:

$$f\langle g,a,b\rangle(x_t, y_t, y_{t-1}) = \begin{cases} 1 & \text{if } g(x_t) = 1 \text{ \& } y_{t-1} = a \text{ \& } y_t = b, \\ 0 & \text{otherwise}. \end{cases}$$

One ”extreme” case is that when for any given input feature g, for each valid combinations of tags $\langle a,b\rangle$, the above defined compound feature $f\langle g,a,b\rangle$ is included in the model. This is a maximum entropy model in which the same set of input features g is associated with transitions between any two hidden states a and b. It can be shown that this type of model is in fact equivalent with a Maximum Entropy Markov Model (MEMM) (McCallum et al., 2000), which means that the same generic system that is currently used for learning a MaxEnt model, can also be used for learning an MEMM model by simply providing it with the appropriate set of features.

An MEMM (McCallum et al., 2000) creates an maximum entropy model for each state in the model. Thus, for a given state s', the framework learns an exponential model corresponding to the probability of transitioning to another state s from s', given the observation sequence o, i.e. $p(s|s', o)$. Consequently, if

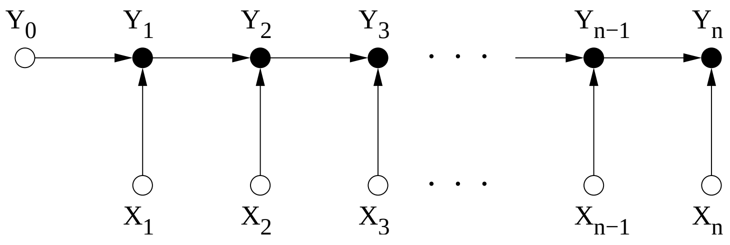

Figure 5: Unrolling an MEMM as a directed graphical model

the set of states is S, the MEMM will contain $|S|$ exponential models. Finding the most likely sequence of states in this context can be done efficiently using a Viterbi-like algorithm. The procedure for learning the parameters is the same as in the MaxEnt case i.e. using Improved Iterative Scaling (Della Pietra, Della Pietra, & Lafferty, 1997) or a gradient based method (the likelihood function is concave, and the gradient is simply the difference between observed and expected feature counts).

2.2.4 Conditional Random Fields

A fundamental problem with MEMMs and other discriminative Markov models based on directed graphical models is that they are biased toward states with few successor states. This is the ”label bias problem” (Lafferty et al., 2001), which in a more general form stipulates that states with low entropy next-state distributions will take little notice of observations. The maximum entropy model from (Ratnaparkhi, 1996) is subject to this problem too, as some of the features it uses are indirectly associated with transitions (they contain conditions on labels of consecutive tokens). The reason for this behavior stems from the fact that the same probability mass is allocated for modeling the labeling decision at each position in the sequence. A principled solution to this problem is that of Conditional Random Fields (Lafferty et al., 2001), where a single probability distribution is learned, one that models the joint probability of a label sequence given a sequence of observations. Informally, this can be viewed as a finite state model with unnormalized transition probabilities. Therefore, some transitions may contribute more than others to the overall score, depending on the corresponding observations.

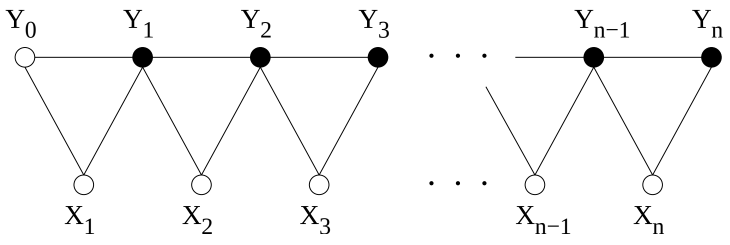

Figure 6: Unrolling a CRF as an undirected graphical model

Inference in CRFs can be done efficiently by accommodating the corresponding forward-backward or Viterbi algorithms used for HMMs (Rabiner, 1989). Learning the CRFs parameters can be cast as an optimization problem – the likelihood function is concave, thus a global maximum can be found efficiently using standard procedures, such as Improved Iterative Scaling (Della Pietra et al., 1997), or gradient based methods.

We have started the list of token classification approaches with HMM models, which are generative and can be represented as directed graphical models. We have argued that conditional models are more appropriate for the tagging task, one of their benefits being that they allow for arbitrary, potentially overlapping features over the observation sequence. Consequently, we have described Maximum Entropy models, a class of conditional models which we have further shown that it subsumes Maximum Entropy Markov Models, a particular type of conditional Markov models. Although these conditional models offer increased representational power when compared with HMMs (their generative counterpart), they are all plagued by the ”label bias problem”. This is particularly troublesome, as the problem does not occur with HMMs. The solution came in the form of Conditional Random Fields, a type of undirected graphical models especially suited for labeling sequences, which overcomes the label bias problem by modeling the joint probability over the entire label sequence given the observation sequence. In the next sections we describe a generic type of undirected graphical models called Relational Markov Networks (RMNs) (Taskar et al., 2002) which can model more general types of label correlations, and are consequently a suitable framework for our initial approach to ”collective information extraction”.

2.3 Markov Random Fields

Graphical models offer an intuitive representation of conditional independence between domain variables. They come in two main flavors:

- **Directed Models** – well suited to represent temporal and causal relationships (Bayesian Networks, Neural Networks, HMMs)
- **Undirected Models** – appropriate for representing statistical correlations between variables (Markov Networks such as CRFs, RMNs, Boltzman Machines)

Markov Random Fields (Markov Networks) are a special class of undirected graphical models. Below is their definition, based on the following notation:

- V = a set of vertices used to denote random variables
- $G = (V, E)$ an undirected graph
- $N(v)$ = the set of neighbors of vertex $v \in V$

Definition 1 *V is said to be a Markov Random Field with respect to G if for any vertex, its value depends only on its neighbors i.e.* $P(V_i|V - V_i) = P(V_i|N(V_i)), \forall V_i \in V$

For the discriminative version, assume X is the set of observed variables, and Y is the set of hidden variables, such that $V = X \cup Y$.

Definition 2 *V is said to be a Conditional Markov Random Field with respect to G if* $P(Y_i|X, Y - Y_i) = P(Y_i|X, N(Y_i)), \forall Y_i \in Y$

Markov Random Fields characterize the underlying undirected graphical model via a local property, namely the Markov assumption. On the other hand, Gibbs Random Fields, which are going to be defined next, use a global property to characterize the corresponding graphical model. The corresponding notation follows below:

- V = a set of vertices which stand for random variables
- $G = (V, E)$ an undirected graph
- $C(G)$ = the set of cliques in G
- V_c = the set of vertices in a clique $c \in C$
- $\phi = \{\phi_c : V_c \rightarrow R_+, c \in C(G)\}$ a set of *clique potentials*

Definition 3 *V is said to be a Gibbs Random Field with respect to G if $P(V) = \frac{1}{Z}\sum_{c \in C(G)} \phi_c(V_c)$, where Z is a normalization constant.*

Thus, a Gibbs Random Field is specified numerically by associating potentials with cliques in the graph. A clique potential is a function on the set of possible configurations of the clique, that associates a positive number with each configuration. The joint probability distribution over all vertices in the graph is obtained by taking a product over the clique potentials.

For the discriminative version, assume X is the set of observed variables, and Y is the set of hidden variables, such that $V = X \cup Y$, and similarly, for every clique $c \in C(G)$, let $V_c = X_c \cup Y_c$.

Definition 4 *V is said to be a Conditional Gibbs Random Field with respect to G if $P(Y|X) = \frac{1}{Z(X)}\sum_{c \in C(G)} \phi_c(X_c, Y_c)$, where $Z(X)$ is a normalization constant.*

Therefore, whereas a Markov Random Field is an undirected graphical model characterized by a local property, a Gibbs Random Field is an undirected graphical model constrained by a global property e.g. the Gibbs distribution. The following theorem stipulates that the two types of graphical models are in fact equivalent.

Theorem 1 *(Hammersley & Clifford, 1971) V is a (conditional) MRF with respect to G if and only if V is a (conditional) GRF with respect to G.*

Consequently, one can create a Markov Random Field by specifying an underlying probability distribution that factorizes into potentials over all maximal cliques in the graph.

2.4 Relational Markov Networks

Relational Markov Networks (Taskar et al., 2002) are conditional Markov random fields augmented with a set of *clique templates*. A clique template specifies which vertices are to be connected in a clique, associating the same clique potential with all cliques that it creates in the graph. Thus, a clique template provides at the same time a procedure for creating edges in the graph, and a mechanism for tying parameters (clique potentials) in the model.

In (Taskar et al., 2002), the RMN framework was introduced in order to model correlations between the class labels of hyperlinked web pages – pages which are hyperlinked tend to have the same label. The clique template responsible for this type of correlations is detailed below:

- **Clique Creation** Add an edge (a 2-node clique) between the labels of any two hyperlinked web pages.
- **Clique Potentials** To all edges created by this template, associate the same potential function ϕ. If the number of possible class labels is N, then ϕ can be specified as an $N \times N$ table of positive real values i.e. $\phi : \{1, 2, ..., N\} \times \{1, 2, ..., N\} \rightarrow R_+$.

Figure 7 shows a sample RMN, where the above clique template creates eight edges between the labels Y of hyperlinked web pages X. The same potential ϕ is associated with all these edges. Other clique templates are responsible for creating edges between each label Y_i and the corresponding local context features in X_i.

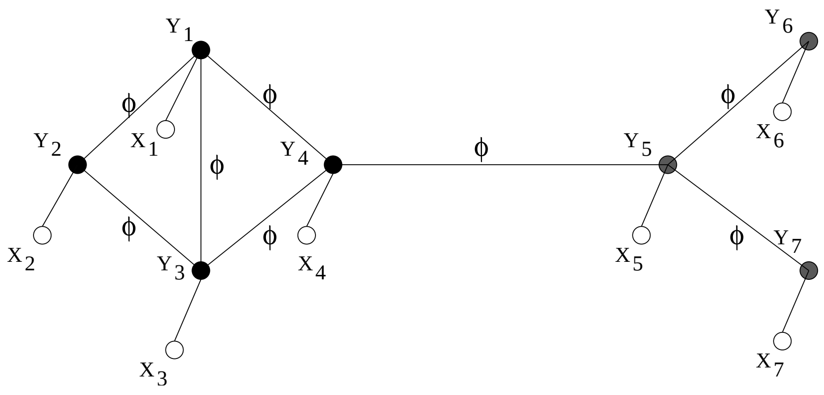

Figure 7: An RMN unrolled, with cliques between hyperlinked web pages.

The CRFs, as previously illustrated in Figure 6, are therefore a particular type of RMNs, in which clique templates create 3-node cliques between any two consecutive labels, Y_{t-1} and Y_t, and their corresponding contextual features X_t.

Given a set of potentials, doing inference with RMNs may refer to two things:

1. Computing the marginal probabilities for all hidden variables, or a proper subset of them.

2. Computing the most probable assignment of values to all hidden variables in the model.

For tree-structured models, the belief propagation algorithm (Pearl, 1988) computes the marginals over all hidden variables in time linear with the number of nodes and edges in the underlying graph. For graphs with cycles, however, exact inference algorithms, such as the join-tree algorithm, have a running time exponential in the size of the largest clique in the triangulated graph. An alternative to exact inference is to do approximate inference using loopy belief propagation, which has shown reasonable performance in many practical applications (Murphy et al., 1999).

Learning with RMNs means computing the clique potential for each potential template, given training data where both the content attributes and the labels are observed. One alternative is to use a gradient based method in a Maximum Likelihood (ML) or Maximum A Posteriori (MAP) setting. For the last type of estimation, a “shrinkage” prior over the parameters is used, typically a zero-mean Gaussian. Because, in both cases, the objective function is concave, the optimization procedure is guaranteed to find a global maximum. An alternative learning method is to use *stochastic gradient ascent* in the form of a Voted Perceptron (Collins, 2002). In this case, the objective function is calculated for a single instance at a time, and its gradient is approximated with the features counts on the Most Probable Explanation (MPE) labeling, instead of computing the full feature count expectation. Nevertheless, inference is needed in both learning scenarios, either for computing marginals over subsets of hidden variables, of for deriving the MPE labeling.

Viewed from the RMN perspective, CRFs are a special type of linear-chain undirected graphical models, and, as with any linear-chain or tree-structured graphical models, both exact inference and parameter estimation can be solved efficiently.

3 Completed Research

Of all IE systems mentioned in the previous section, that of (Seymore et al., 1999) is able to model influences between various types of entities based on the order in which they occur in the document – in headers of research papers, for example, the author's name usually comes after the title. This type of order-based correlations is captured by learning an HMM structure in which the same entity type may be associated with multiple states in the model, while the set of transitions reflects the order in which various entity types occur in the training data.

There have been some previous attempts to use global information from repetitions, acronyms, and abbreviations during extraction. In (Chieu & Ng, 2003), a set of global features are used to improve a Maximum-Entropy tagger; however, these features do not fully capture the mutual influence between the labels of acronyms and their long forms, or between entity repetitions. In particular, they only allow earlier extractions in a document to influence later ones and not vice-versa.

In this section we are going to introduce a collective approach to Information Extraction which will allow the incorporation of arbitrary correlations between the labels of potential extractions from the same document. For this, we shall use the RMN framework to do extraction by phrase classification.

3.1 Candidate Entities

Doing phrase classification requires a set of phrases to start with. Throughout this document, we will use the terms *candidate entities*, *candidate extractions*, or *candidate phrases* to refer to the set of phrases that are to be classified as being valid extractions or not. 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 program. Various heuristics exist however which can significantly reduce the size of the candidate set, and some of them 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 whose POS tags are from {*"JJ", "VBN", "VBG", "POS", "NN", "NNS", "NNP", "NNPS", "CD", "–"*}, and whose last word (the head) is tagged either as a noun, or a number. Candidate extractions then consist of base NPs, together with all their contiguous subsequences headed by a noun or number.

- **H3:** The CoNLL 2003 English corpus (Tjong Kim Sang & De Meulder, 2003) contains four types of named entities: persons (PER), locations (LOC), organizations (ORG), and other (MISC). A more appropriate heuristic in this case is to consider as candidates all sequences of proper names, potentially interspersed with prepositions, commas, conjunctions or definite articles.

Table 1 below shows the candidate entities generated by H1 and H2 on a fragment from a Medline abstract. Similarly, Table 2 shows candidate entities generated by H1 and H3 on a fragment from a CoNLL document. Both H2 and H3 are strong heuristics, in the sense that they drastically reduce the number of candidate entities. In the next sections, we shall focus on the task of extracting protein names from Medline abstracts.

	"the control of human **ribosomal protein L22** *(* **rpL22** *)* "
H1	∘ the ∘ the control ∘ the control of ∘ the control of human ∘ the control of human ribosomal ∘ ... ∘ ribosomal ∘ ribosomal protein ∘ **ribosomal protein L22** ∘ ribosomal protein L22 (∘ ribosomal protein L22 (rpL22 ∘ ... ∘ L22 ∘ L22 (∘ ∘ L22 (rpL22 ∘ L22 (rpL22) ∘ ... ∘ **rpL22** ∘ rpL22) ∘) ∘
H2	∘ control ∘ human ribosomal protein ∘ human ribosomal protein L22 ∘ ribosomal protein ∘ **ribosomal protein L22** ∘ protein L22 ∘ L22 ∘ **rpL22** ∘

Table 1: Candidate Extractions: Medline.

	"**Israel** *gave* **Palestinian** *President* **Yasser Arafat** *permission on Thursday*"
H1	∘ **Israel** ∘ Israel gave ∘ Israel gave Palestinian ∘ Israel gave Palestinian President ∘ ... ∘ **Palestinian** ∘ Palestinian President ∘ Palestinian President Yasser ∘ Palestinian President Yasser Arafat ∘ ... ∘ Yasser ∘ **Yasser Arafat** ∘ President Yasser Arafat permission ∘ ... ∘ on ∘ on Thursday ∘ Thursday ∘
H3	∘ **Israel** ∘ **Palestinian** ∘ Palestinian President ∘ Palestinian President Yasser ∘ Palestinian President Yasser Arafat ∘ President ∘ President Yasser ∘ President Yasser Arafat ∘ Yasser ∘ **Yasser Arafat** ∘ Arafat ∘

Table 2: Candidate Extractions: CoNLL.

3.2 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 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 3, 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.

3.3 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 (Section 3.1). Each entity $e \in d.E$ is

Description	Feature Template
Head Word	$w_{(n)}$
Text	$w_{(0)}$-$w_{(1)}$-$\cdots$-$w_{(n)}$
Short Type	$s_{(0)}$-$s_{(1)}$-$\cdots$-$s_{(n)}$
Bigram Left (4 bigrams)	$w_{(-1)}$-$w_{(0)}$ $w_{(-1)}$-$s_{(0)}$ $s_{(-1)}$-$w_{(0)}$ $s_{(-1)}$-$s_{(0)}$
Bigram Right (4 bigrams)	$w_{(n)}$-$w_{(n+1)}$ $w_{(n)}$-$s_{(n+1)}$ $s_{(n)}$-$w_{(n+1)}$ $s_{(n)}$-$s_{(n+1)}$
Trigram Left (8 trigrams)	$w_{(-2)}$-$w_{(-1)}$-$w_{(0)}$ $\cdots$ $s_{(-2)}$-$s_{(-1)}$-$s_{(0)}$
Trigram Right (8 trigrams)	$w_{(n)}$-$w_{(n+1)}$-$w_{(n+2)}$ $\cdots$ $s_{(n)}$-$s_{(n+1)}$-$s_{(n+2)}$
POS Left	$t_{(-1)}$
POS Right	$t_{(n+1)}$
Prefix (n+1 prefixes)	$s_{(0)}$ $s_{(0)}$-$s_{(1)}$ $\cdots$ $s_{(0)}$-$s_{(1)}$-$\cdots$-$s_{(n+1)}$
Suffix (n+1 suffixes)	$s_{(n)}$ $s_{(n-1)}$-$s_{(n)}$ $\cdots$ $s_{(0)}$-$s_{(1)}$-$\cdots$-$s_{(n+1)}$

Table 3: Feature Templates.

characterized by a predefined set of boolean attributes $e.F$ (Section 3.2), 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.

Each document is associated with an undirected graphical model, with nodes corresponding directly to entity attributes, one node for each attribute of each candidate entity in the document. The set of edges is created by matching *clique templates* against the entire set of 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 a selected set of attribute nodes so that they form a clique.

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 3.4 and 3.5 where the clique templates used in our model are described in detail.

Depending on the number of hidden labels in Y_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 graph model for a document d has been completed with cliques 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:

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

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

$$Z(d.X) = \sum_{Y} \prod_{c \in C} \prod_{G \in M_c(d.E)} \phi_C(G.X_c, G.Y_c) \tag{2}$$

3.4 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 3) and its label. If, after filtering, we are left with h distinct boolean features f_i, one way to model these correlations is to introduce h local (clique) templates $LT_1, LT_2, ..., LT_h$. A template LT_i would then be defined as follows:

1. The matching operator M_i is set to match any single-entity set $\{e\}$.
2. The collection of attributes S_i corresponding to a singleton entity set $\{e\}$ is defined to be $S_i = \langle X_i, Y_i \rangle = \langle \{e.f_i\}, \{e.label\} \rangle$. This amounts to introducing in the RMN graph h attribute nodes for each candidate entity, which are to be connected by the h local templates to the corresponding entity label node. The 2-node cliques created by all h templates around one entity are illustrated in Figure 8.
3. The potential ϕ_i associated with all 2-node cliques created by template LT_i would consist in a 2×2 table (as both $e.f_i$ and $e.label$ have cardinality 2 – assuming only one entity type is to be extracted, we need only two values for the label attribute).

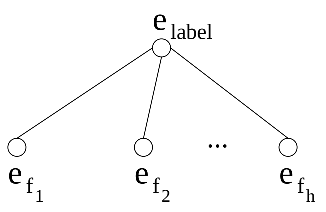

Figure 8: RMN generated by local templates.

Each entity has the label node connected to its own set of h binary feature nodes. This leads to an excessive number of nodes in the model, most of which have value zero. The number of nodes can be reduced if, for each entity, we include in the graphical model only those nodes for which the corresponding feature variable has value 1. Consequently, the table associated with the local potential will be reduced from 4 to 2 values, specifying now the compatibility between that feature and the two possible values for the entity label.

Factor Graphs. An alternative, useful representation for Markov random fields is provided by factor graphs (Kschischang, Frey, & Loeliger, 2001). These are bipartite graphs which express how a global function of many variables (the probability $P(d.Y|d.X)$ in Equation 1) factors into a product of local functions (the potentials $\phi_C(G.X_c, G.Y_c)$ in Equation 1). Factor graphs subsume many different types of graphical models, including Bayesian networks and Markov random fields. The sum/max-product algorithm used for inference in factor graphs generalizes a wide variety of algorithms including the forward/backward algorithm, the Viterbi algorithm, and Pearl's belief propagation algorithm (Pearl, 1988). To obtain the factor graph for a given Markov random field, we copy all original nodes from the MRF, referred henceforth as *variable nodes*, and create a *potential node* for each instantiated clique potential. Each potential node is then linked to all variable nodes from the associated clique.

In the case of local clique potentials, given that all feature nodes have value 1, we can eliminate them from the equivalent factor graph representation. What is left then is a variable node for the entity label, together with nodes for potential functions, one potential node for each entity feature whose value has been observed to be 1. As an example, Figure 9 shows that part of the factor graph which is generated around the entity label for 'HDAC1 enzyme' (with variable nodes figured as empty circles and potential nodes figured as black squares).

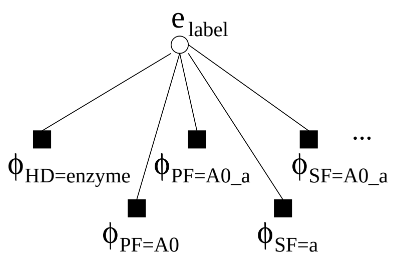

Figure 9: Factor Graph for local templates.

Note that the factor graph above has an equivalent RMN graph consisting of a one-node clique only, on which it is hard to visualize the various potentials involved. There are cases where different factor graphs may yield the same underlying RMN graph, which makes the factor graph representation preferable.

3.5 Global Clique Templates

Global clique templates enable us to model hypothesized influences between entities from the same document. They connect the label nodes of two or more entities, which, in the factor graph, translates into potential nodes connected to at least two label nodes. 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). Later 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.

3.5.1 The Overlap Template

The definition of a *candidate extraction* from Section 3.1 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 as follows:

1. The M_{OT} operator matches any two overlapping candidate entities $\{e_1, e_2\}$.

2. The set of attributes S_{OT} selected by this template for two overlapping entities $\{e_1, e_2\}$ is $S_{OT} = \langle X_{OT}, Y_{OT} \rangle = \langle \emptyset, \{e_1.label, e_2.label\} \rangle$. This translates in the factor graph into a potential node connected to the two selected label nodes.

3. The potential function ϕ_{OT} is set so that at most one of the overlapping entities can have label-value 1, as illustrated in Table 4.

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

Table 4: Overlap Potential.

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.

An alternative solution for the overlap template is to create a potential node for each token position that is covered by at least two candidate entities in the document, and connect it to their label nodes. The difference in this case is that the potential node will be connected to a variable number of entity label nodes. However this second approach has the advantage of creating fewer potential nodes in the document factor graph, which results in faster inference.

3.5.2 The Repeat Template

We could specify the potential for the repeat template in a similar 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 3.5.1, the candidate entity 'glutathione' is included in two other entities: 'glutathione S' and 'glutathione S - transferase'.

In order to instantiate potentials over 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 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 cliques are only created as needed, e.g. when the auxiliary OR entity is required by repeat and acronym clique templates.

Figure 10 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. To avoid clutter, all entities in this and subsequent factor graphs stand for their corresponding label features. The potential function ϕ_{RT} can either be 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.

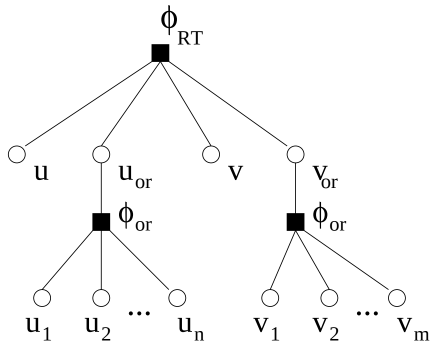

Figure 10: Repeat Factor Graph.

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, while the including entities will be v_1 = 'glutathione antioxidant' and v_2 = 'glutathione antioxidant system'.

3.5.3 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 in (Schwartz & 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 ϕAT, as illustrated in Figure 11.

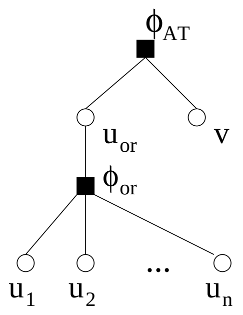

Figure 11: Acronym Factor Graph.

For example, consider the phrase 'the antioxidant superoxide dismutase - 1 (SOD1)', where both 'superoxide dismutase - 1' and 'SOD1' are tagged as proteins. 'SOD1' satisfies our criteria for acronyms, thus it will be associated with the entity v in Figure 11. The candidate long forms are u_1 = 'antioxidant superoxide dismutase - 1', u_2 = 'superoxide dismutase - 1', and u_3 = 'dismutase - 1'.

3.6 Inference in Factor Graphs

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.

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:

$$d.Y^* = arg \max_{d.Y} P(d.Y|d.X) \tag{3}$$

where $P(d.Y|d.X)$ is defined as in Equation 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