Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2
pp. 439-446, Barcelona, Spain, July 2004.

Collective Information Extraction with Relational Markov Networks

Razvan Bunescu
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712
razvan@cs.utexas.edu

Raymond J. Mooney
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712
mooney@cs.utexas.edu

Abstract

Most information extraction (IE) systems treat separate potential extractions as independent. However, in many cases, considering influences *between* different potential extractions could improve overall accuracy. Statistical methods based on *undirected* graphical models, such as *conditional random fields* (CRFs), have been shown to be an effective approach to learning accurate IE systems. We present a new IE method that employs Relational Markov Networks (a generalization of CRFs), which can represent arbitrary dependencies between extractions. This allows for "collective information extraction" that exploits the mutual influence between possible extractions. Experiments on learning to extract protein names from biomedical text demonstrate the advantages of this approach.

1 Introduction

Information extraction (IE), locating references to specific types of items in natural-language documents, is an important task with many practical applications. 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). One of the current best empirical approaches to IE is *conditional random fields* (CRF's) (Lafferty et al., 2001). CRF's are a restricted class of *undirected graphical models* (Jordan, 1999) designed for sequence segmentation tasks such as IE, part-of-speech (POS) tagging (Lafferty et al., 2001), and shallow parsing (Sha and Pereira, 2003). In a recent follow-up to previously published experiments comparing a large 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., 2004), CRF's were found to significantly out-perform competing techniques.

As typically applied, CRF's, like almost all IE methods, assume separate extractions are independent and treat each potential extraction in isolation. However, in many cases, considering influences *between* extractions can be very useful. 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 indicative of protein references.

Relational Markov Networks (RMN's) (Taskar et al., 2002) are a generalization of CRF's that allow for *collective classification* of a set of related entities by integrating information from features of individual entities as well as the relations between them. Results on classifying connected sets of web pages have verified the advantage of this approach (Taskar et al., 2002). In this paper, we present an approach to *collective information extraction* using RMN's that simultaneously extracts all of the information from a document by exploiting the textual content and context of each relevant substring as well as the document relationships between them. Experiments on human protein tagging demonstrate the advantages of collective extraction on several annotated corpora of Medline abstracts.

2 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. Each entity $e \in d.E$ is characterized by a predefined set of boolean features $e.F$. This set of features is the same for all candidate entities, and it can be assimilated with the relational database definition of a table. One particular feature 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 features, 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 features, one node for each feature 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 feature 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.
2. A selected set of features $S_c = \langle X_c, Y_c \rangle$ for 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 that 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, $M_c(E) \subseteq 2^E$ consists of subsets of entities whose feature 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, our clique templates may match a variable number of entities. The set S_c may contain the same feature from different entities. Usually, for each entity in the matching set, its label is included in S_c. All these will be illustrated with examples in Sections 4 and 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 computed as:

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

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

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

The above distribution presents the RMN as a Markov random field (MRF) with the clique templates as a method for tying potential values across different cliques in the graphical model.

3 Candidate Entities and Entity Features

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.

Definition 1: A *base noun phrase* is 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 consist of base NPs, augmented with all their contiguous subsequences headed by a noun or number.

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 lowercase 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, where we consider a generic candidate extraction as a sequence of $n+1$ words $w_0 w_1 \ldots w_n$.

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 1: Feature Templates.

4 Local Clique Templates

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 are filtered out. If, after filtering, we are left with h distinct boolean features $(f_i{=}v_j)$, we create h local (clique) templates $LT_1, LT_2, ..., LT_h$. Each template's matching operator is set to match any single-entity set. The collection of features S_i corresponding to template LT_i applied to the singleton entity set $\{e\}$ is $S_i = \langle X_i,\ Y_i\rangle = \langle\{\ e.f_i{=}v_j\},\ \{e.label\}\rangle$. The 2-node cliques created by all h templates around one entity are illustrated in Figure 1.

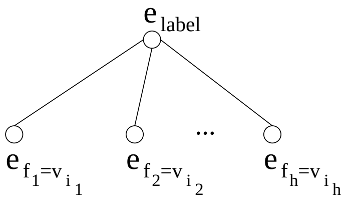

Figure 1: RMN generated by local templates.

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 the value zero. To reduce the number of nodes, we could remove the observed nodes from the graph, which then results in many one-node clique potentials (corresponding to the observed features) being associated with the same label node. Because these clique potentials may no longer be distinguished in the RMN graph, in order to have all of them as explicit as possible in the graphical model, we transform the relational Markov network into its equivalent *factor graph* representation. Factor graphs (Kschischang et al., 2001) are bipartite graphs that 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 *feature nodes* from the MRF, and create a *potential node* for each instantiated clique potential. Each potential node is then linked to all nodes from the associated clique. However in this case, instead of creating a potential node for each feature-value pair as in the initial MRF model, we create a potential node only for the binary features that are 1 for the given entity. Correspondingly, the table associated with the potential will be reduced from 4 to 2 values. As an example, Figure 2 shows that part of the factor graph which is generated around the entity label for HDAC1 enzyme (with feature nodes figured as empty circles and potential nodes figured as black squares).

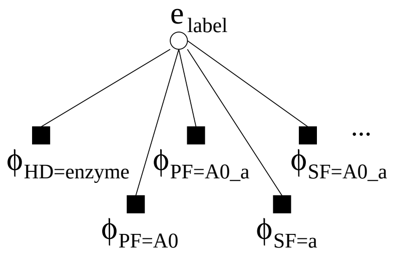

Figure 2: Factor Graph for local templates.

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.

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.

5.1 The Overlap Template

The definition of a *candidate extraction* from Section 3 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, because the other four entities overlap with it, they should all have label-value 0.

This type of constraint is enforced by the overlap template whose M operator matches any two overlapping candidate entities, and which connects their label nodes (specified in S) through a potential node with a potential function ϕ that allows at most one of them to have label-value 1, as illustrated in Table 2. 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.

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

Table 2: Overlap Potential.

5.2 The Repeat Template

We could specify the potential for the repeat template in a similar 2-by-2 table, this time leaving the table entries to be learned, given that it 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 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 feature $e_{or}.label$. The potential function 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 variable is required by repeat and acronym clique templates.

Figure 3 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 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.

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'.

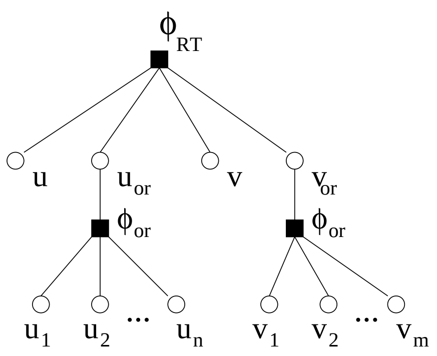

Figure 3: Repeat Factor Graph.

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 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 variable u_{or}, and connect it to v's node label through an acronym potential, as illustrated in Figure 4. 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 4. The candidate long forms are u_1 = 'antioxidant superoxide dismutase - 1', u_2 = 'superoxide dismutase - 1', and u_3 = 'dismutase - 1'.

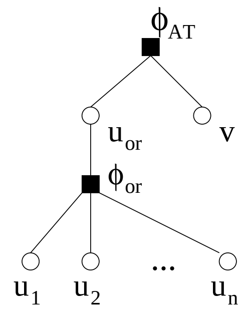

Figure 4: Acronym Factor Graph.

6 Inference in Factor Graphs

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:

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

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 belief at each label node by repeatedly passing messages between the node and all potential nodes connected to it (Kschischang et al., 2001). As many of the label nodes are indirectly connected through potential nodes instantiated by global templates, their belief values will propagate in the graph and mutually influence each other, leading in the end to a collective labeling decision.

The time complexity of computing messages from a potential node to a label node is exponential in the number of label nodes attached to the potential. Since this "fan-in" can be large for OR potential nodes, this step required optimization. Fortunately, due to the special form of the OR potential, and the normalization before each message-passing step, we were able to develop a linear-time algorithm for this special case. Details are omitted due to limited space.

7 Learning Potentials in Factor Graphs

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

$$\phi_C(G.X_c, G.Y_c) = exp\{\mathbf{w_c f_c}(G.X_c, G.Y_c)\} \quad (4)$$

where f_c is a vector of binary features, one for each configuration of values for X_c and Y_c.

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 $\mathbf{w}$. This expectation 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 use Collins' voted perceptron approach (Collins, 2002), which approximates the full expectation of $\mathbf{f_c}$ with the $\mathbf{f_c}$ counts for the most likely labeling under the current parameters, $\mathbf{w}$. In all our experiments, the perceptron was run for 50 epochs, with a learning rate set at 0.01.

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. Of these, 147 have been randomly selected by posing a query containing the (Mesh) terms *protein binding*, *interaction*, and *molecular* to Medline, while the rest of 53 have been extracted randomly from the GENIA corpus (Collier et al., 1999). It contains a total of 3713 protein references. The second dataset is Aimed[2] which has been previously used for training the protein interaction extraction systems in (Bunescu et al., 2004). It consists of 225 Medline abstracts, of which 200 are known to describe interactions between human proteins, while the other 25 do not refer to any interaction. There are 4084 protein references in this dataset. 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 both local and global templates, and **CRF**, which uses a CRF for labeling token sequences. We used the CRF implementation from (McCallum, 2002) with the set of tags and features used by the Maximum-Entropy tagger described in (Bunescu et al., 2004). All Medline abstracts were tokenized and then POS tagged using Brill's tagger (Brill, 1995). 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 Tables 3 and 4 which give average precision, recall, and F-measure using 10-fold cross validation.

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

Table 3: Extraction Performance on Yapex.

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

Table 4: Extraction Performance on Aimed.

These tables show that, in terms of F-measure, the use of global templates for mod-

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

eling 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 CRF's, with the results being statistically significant only for the Yapex dataset, corresponding to a p value of 0.02. We hypothesize that further improvements to the LT-RMN approach would push the GLT-RMN performance even higher. The tagging scheme used by CRFs, in which each token is assigned a tag, is essentially different from the RMN approach, where candidate extractions are either rejected or accepted. In the tagging approach used by CRFs, extracted entities are available only after tagging is complete, thereby making it difficult to account for influences between them during tagging.

Figures 5 and 6 show the precision-recall curves for the two datasets. These were obtained by varying a threshold on the extraction confidence, which is the posterior probability that its label is 1 as computed by the sum-product algorithm.

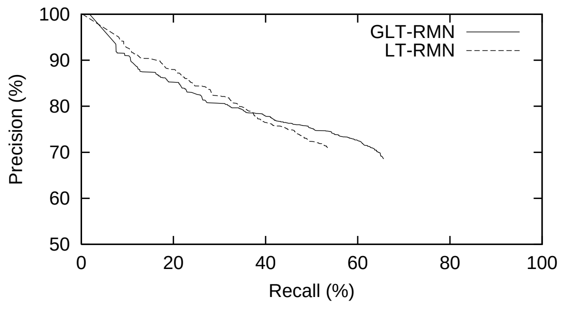

Figure 5: Precision Recall Curves on Yapex.

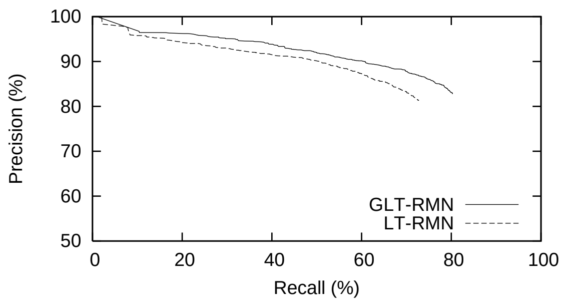

Figure 6: Precision Recall Curves on Aimed.

We also explored using a global template that captured the tendency for candidate entities whose phrases are coordinated to have the same label. This technique did not improve performance since detecting whether two NPs are coordinated is difficult, and the methods we tried introduced too many false coordinations.

In order to evaluate the applicability of our method to other types of narrative, we also tried it on the CoNLL 2003 English corpus (Tjong Kim Sang and De Meulder, 2003) which contains four types of named entities: persons (PER), locations (LOC), organizations (ORG), and other (MISC). Consequently the number of label values increased from two to five (with a label-value of 0 to indicate none of the four categories). For the global approach we used the same overlap template and a modified version of the repeat template in which the OR potential was replaced with a different type of potential (SEL) that allows at most one of the including entities to have a non-zero label-value. The SEL variable (replacing the OR variable) is forced to have label-value 0 if all including entities have label-value 0, otherwise it selects the one label-value that is not 0. The resulting repeat template, besides handling exact repetitions, is also able to capture correlations between entity types, when one entity repetition is included in another entity with a potentially different type. For example, it is common in this corpus to have country names repeated inside organization names in the same document, as is "Japan" in "Bank of Japan", or "Japan Aluminium Federation".

The overall results are shown in Table 5, with the global approach exhibiting improvement over the local approach, albeit less pronounced than in the biomedical domain. No dictionaries were used in these experiments, and no custom feature selection was performed – the feature templates were the same as those used in the biomedical extraction.

Method	Precision	Recall	F-measure
LT-RMN	82.15	78.13	80.09
GLT-RMN	83.17	81.44	82.30
CRF	81.57	80.08	80.82

Table 5: Extraction Performance on CoNLL.

9 Related Work

There have been some previous attempts to use global information from repetitions, acronyms, and abbreviations during extraction. In (Chieu and 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. The RMN approach handles these and potentially other mutual influences between entities in a more complete, probabilistically sound manner.

10 Conclusions and Future Work

We have presented an approach to collective information extraction that uses Relational Markov Networks to reason about the mutual influences between multiple extractions. A new type of clique template – the logical OR template – was introduced, allowing a variable number of relevant entities to be used by other clique templates. Soft correlations between repetitions and acronyms and their long form in the same document have been captured by global clique templates, allowing for local extraction decisions to propagate and mutually influence each other.

Regarding future work, a richer set of features for the local templates would likely improve performance. Currently, LT-RMN's accuracy is still significantly less than CRF's, which limits the performance of the full system. Another limitation is the approximate inference used by both RMN methods. The number of factor graphs for which the sum-product algorithm did not converge was non-negligible, and our approach stopped after a fix number of iterations. Besides exploring improvements to loopy belief propagation that increase computational cost (Yedidia et al., 2000), we intend to examine alternative approximate-inference methods.

11 Acknowledgements

This work was partially supported by grants IIS-0117308 and IIS-0325116 from the NSF.

References

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

Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun Kumar Ramani, and Yuk Wah Wong. 2004. Comparative experiments on learning information extractors for proteins and their interactions. *Special Issue in the Journal Artificial Intelligence in Medicine on Summarization and Information Extraction from Medical Documents.* To appear.

Mary Elaine Califf, editor. 1999. *Papers from the AAAI-1999 Workshop on Machine Learning for Information Extraction*, Orlando, FL. AAAI Press.

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

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003)*, pages 160–163, Edmonton, Canada.

N. Collier, H. Park, N. Ogata, Y. Tateisi, C. Nobata, T.Ohta, T. Sekimizu, H. Imai, K. Ibushi, and J. Tsujii. 1999. The GENIA project: Corpus-based knowledge acquisition and information extraction from genome research papers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL-99)*, pages 271–272, Bergen.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 489–496, Philadelphia, PA.

Michael I. Jordan, editor. 1999. *Learning in Graphical Models.* MIT Press, Cambridge, MA.

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

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. 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, Williams College, MA.

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

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo,CA.

Ariel S. Schwartz and Marti A. Hearst. 2003. 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.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology and the Meeting of the North American Association for Computational Linguistics*, pages 134–141, Edmonton, Canada.

Benjamin Taskar, Pieter Abbeel, and D. Koller. 2002. Discriminative probabilistic models for relational data. In *Proceedings of 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 485–492, Edmonton, Canada.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. 2000. Generalized belief propagation. In *Advances in Neural Information Processing Systems 12*, pages 689–695, Denver, CO.