Simple Algorithms for Complex Relation Extraction with Applications to Biomedical IE

Ryan McDonald[1] **Fernando Pereira**[1] **Seth Kulick**[2]
[1]CIS and [2]IRCS, University of Pennsylvania, Philadelphia, PA
{ryantm,pereira}@cis.upenn.edu, skulick@linc.cis.upenn.edu

Scott Winters **Yang Jin** **Pete White**
Division of Oncology, Children's Hospital of Pennsylvania, Philadelphia, PA
{winters,jin,white}@genome.chop.edu

Abstract

A complex relation is any n-ary relation in which some of the arguments may be be unspecified. We present here a simple two-stage method for extracting complex relations between named entities in text. The first stage creates a graph from pairs of entities that are likely to be related, and the second stage scores maximal cliques in that graph as potential complex relation instances. We evaluate the new method against a standard baseline for extracting genomic variation relations from biomedical text.

1 Introduction

Most research on text information extraction (IE) has focused on accurate tagging of named entities. Successful early named-entity taggers were based on finite-state generative models (Bikel et al., 1999). More recently, discriminatively-trained models have been shown to be more accurate than generative models (McCallum et al., 2000; Lafferty et al., 2001; Kudo and Matsumoto, 2001). Both kinds of models have been developed for tagging entities such as people, places and organizations in news material. However, the rapid development of bioinformatics has recently generated interest on the extraction of biological entities such as genes (Collier et al., 2000) and genomic variations (McDonald et al., 2004b) from biomedical literature.

The next logical step for IE is to begin to develop methods for extracting meaningful relations involving named entities. Such relations would be extremely useful in applications like question answering, automatic database generation, and intelligent document searching and indexing. Though not as well studied as entity extraction, relation extraction has still seen a significant amount of work. We discuss some previous approaches at greater length in Section 2.

Most relation extraction systems focus on the specific problem of extracting binary relations, such as the *employee of* relation or *protein-protein interaction* relation. Very little work has been done in recognizing and extracting more complex relations. We define a *complex relation* as any n-ary relation among n typed entities. The relation is defined by the *schema* $(t_1, \ldots, t_n)$ where $t_i \in T$ are entity types. An *instance* (or *tuple*) in the relation is a list of entities $(e_1, \ldots, e_n)$ such that either type$(e_i) = t_i$, or $e_i = \perp$ indicating that the ith element of the tuple is missing.

For example, assume that the entity types are $T = \{\mathsf{person}, \mathsf{job}, \mathsf{company}\}$ and we are interested in the ternary relation with schema $(\mathsf{person}, \mathsf{job}, \mathsf{company})$ that relates a person to their job at a particular company. For the sentence *"John Smith is the CEO at Inc. Corp."*, the system would ideally extract the tuple $(\textit{John Smith}, \textit{CEO}, \textit{Inc. Corp.})$. However, for the sentence *"Everyday John Smith goes to his office at Inc. Corp."*, the system would extract $(\textit{John Smith}, \perp, \textit{Inc. Corp.})$, since there is no mention of a job title. Hence, the goal of complex relation extraction is to identify all instances of the relation of interest in some piece of text, including

Proceedings of the 43rd Annual Meeting of the ACL, pages 491–498,
Ann Arbor, June 2005.

incomplete instances.

We present here several simple methods for extracting complex relations. All the methods start by recognized pairs of entity mentions, that is, binary relation instances, that appear to be arguments of the relation of interest. Those pairs can be seen as the edges of a graph with entity mentions as nodes. The algorithms then try to reconstruct complex relations by making tuples from selected maximal cliques in the graph. The methods are general and can be applied to any complex relation fitting the above definition. We also assume throughout the paper that the entities and their type are known a priori in the text. This is a fair assumption given the current high standard of state-of-the-art named-entity extractors.

A primary advantage of factoring complex relations into binary relations is that it allows the use of standard classification algorithms to decide whether particular pairs of entity mentions are related. In addition, the factoring makes training data less sparse and reduces the computational cost of extraction. We will discuss these benefits further in Section 4.

We evaluated the methods on a large set of annotated biomedical documents to extract relations related to genomic variations, demonstrating a considerable improvement over a reasonable baseline.

2 Previous work

A representative approach to relation extraction is the system of Zelenko et al. (2003), which attempts to identify binary relations in news text. In that system, each pair of entity mentions of the correct types in a sentence is classified as to whether it is a positive instance of the relation. Consider the binary relation *employee of* and the sentence *"John Smith, not Jane Smith, works at IBM"*. The pair $(\textit{John Smith}, \textit{IBM})$ is a positive instance, while the pair $(\textit{Jane Smith}, \textit{IBM})$ is a negative instance. Instances are represented by a pair of entities and their position in a shallow parse tree for the containing sentence. Classification is done by a support-vector classifier with a specialized kernel for that shallow parse representation.

This approach — enumerating all possible entity pairs and classifying each as positive or negative — is the standard method in relation extraction. The main differences among systems are the choice of trainable classifier and the representation for instances.

For binary relations, this approach is quite tractable: if the relation schema is (t_1, t_2), the number of potential instances is $O(|t_1|\,|t_2|)$, where $|t|$ is the number of entity mentions of type t in the text under consideration.

One interesting system that does not belong to the above class is that of Miller et al. (2000), who take the view that relation extraction is just a form of probabilistic parsing where parse trees are augmented to identify all relations. Once this augmentation is made, any standard parser can be trained and then run on new sentences to extract new relations. Miller et al. show such an approach can yield good results. However, it can be argued that this method will encounter problems when considering anything but binary relations. Complex relations would require a large amount of tree augmentation and most likely result in extremely sparse probability estimates. Furthermore, by integrating relation extraction with parsing, the system cannot consider long-range dependencies due to the local parsing constraints of current probabilistic parsers. The higher the arity of a relation, the more likely it is that entities will be spread out within a piece of text, making long range dependencies especially important.

Roth and Yih (2004) present a model in which entity types and relations are classified jointly using a set of global constraints over locally trained classifiers. This joint classification is shown to improve accuracy of both the entities and relations returned by the system. However, the system is based on constraints for binary relations only.

Recently, there has also been many results from the biomedical IE community. Rosario and Hearst (2004) compare both generative and discriminative models for extracting seven relationships between treatments and diseases. Though their models are very flexible, they assume at most one relation per sentence, ruling out cases where entities participate in multiple relations, which is a common occurrence in our data. McDonald et al. (2004a) use a rule-based parser combined with a rule-based relation identifier to extract generic binary relations between biological entities. As in predicate-argument extraction (Gildea and Jurafsky, 2002), each relation is

always associated with a verb in the sentence that specifies the relation type. Though this system is very general, it is limited by the fact that the design ignores relations not expressed by a verb, as the *employee of* relation in *"John Smith, CEO of Inc. Corp., announced he will resign"*.

Most relation extraction systems work primarily on a sentential level and never consider relations that cross sentences or paragraphs. Since current data sets typically only annotate intra-sentence relations, this has not yet proven to be a problem.

3 Definitions

3.1 Complex Relations

Recall that a complex n-ary relation is specified by a schema $(t_1, \ldots, t_n)$ where $t_i \in T$ are entity types. Instances of the relation are tuples $(e_1, \ldots, e_n)$ where either type$(e_i) = t_i$, or $e_i = \perp$ (missing argument). The only restriction this definition places on a relation is that the arity must be known. As we discuss it further in Section 6, this is not required by our methods but is assumed here for simplicity. We also assume that the system works on a single relation type at a time, although the methods described here are easily generalizable to systems that can extract many relations at once.

3.2 Graphs and Cliques

An *undirected graph* $G = (V, E)$ is specified by a set of *vertices* V and a set of *edges* E, with each edge an unordered pair (u, v) of vertices. $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' = \{(u, v) : u, v \in V', (u, v) \in E\}$. A *clique* C of G is a subgraph of G in which there is an edge between every pair of vertices. A *maximal clique* of G is a clique $C = (V_C, E_C)$ such that there is no other clique $C' = (V_{C'}, E_{C'})$ such that $V_C \subset V_{C'}$.

4 Methods

We describe now a simple method for extracting complex relations. This method works by first factoring all complex relations into a set of binary relations. A classifier is then trained in the standard manner to recognize all pairs of related entities. Finally a graph is constructed from the output of this classifier and the complex relations are determined from the cliques of this graph.

a. All possible relation instances

(*John*, *CEO*, *Inc. Corp.*)
(*John*, $\perp$, *Inc. Corp.*)
(*John*, *CEO*, *Biz. Corp.*)
(*John*, $\perp$, *Biz. Corp.*)
(*John*, *CEO*, $\perp$)
(*Jane*, *CEO*, *Inc. Corp.*)
(*Jane*, $\perp$, *Inc. Corp.*)
(*Jane*, *CEO*, *Biz. Corp.*)
(*Jane*, $\perp$, *Biz. Corp.*)
(*Jane*, *CEO*, $\perp$)
($\perp$, *CEO*, *Inc. Corp.*)
($\perp$, *CEO*, *Biz. Corp.*)

b. All possible binary relations

(*John*, *CEO*)
(*John*, *Inc. Corp.*)
(*John*, *Biz. Corp.*)
(*CEO*, *Inc. Corp.*)
(*CEO*, *Biz. Corp.*)
(*Jane*, *CEO*)
(*Jane*, *Inc. Corp.*)
(*Jane*, *Biz. Corp.*)

Figure 1: Relation factorization of the sentence: *John and Jane are CEOs at Inc. Corp. and Biz. Corp. respectively.*

4.1 Classifying Binary Relations

Consider again the motivating example of the (person, job, company) relation and the sentence *"John and Jane are CEOs at Inc. Corp. and Biz. Corp. respectively"*. This sentence contains two people, one job title and two companies.

One possible method for extracting the relation of interest would be to first consider all 12 possible tuples shown in Figure 1a. Using all these tuples, it should then be possible to train a classifier to distinguish valid instances such as (*John*, *CEO*, *Inc. Corp.*) from invalid ones such as (*Jane*, *CEO*, *Inc. Corp.*). This is analogous to the approach taken by Zelenko et al. (2003) for binary relations.

There are problems with this approach. Computationally, for an n-ary relation, the number of possible instances is $O(|t_1|\,|t_2| \cdots |t_n|)$. Conservatively, letting m be the smallest $|t_i|$, the run time is $O(m^n)$, exponential in the arity of the relation. The second problem is how to manage incomplete but correct instances such as (*John*, $\perp$, *Inc. Corp.*) when training the classifier. If this instance is marked as negative, then the model might incorrectly disfavor features that correlate *John* to *Inc. Corp.*. However, if this instance is labeled positive, then the model may tend to prefer the shorter and more compact incomplete relations since they will be abundant in the positive training examples. We could always ignore instances of this form, but then the data would be heavily skewed towards negative instances.

Instead of trying to classify all possible relation instances, in this work we first classify pairs of entities as being related or not. Then, as discussed in Section 4.2, we reconstruct the larger complex relations from a set of binary relation instances.

Factoring relations into a set of binary decisions has several advantages. The set of possible pairs is much smaller then the set of all possible complex relation instances. This can be seen in Figure 1b, which only considers pairs that are consistent with the relation definition. More generally, the number of pairs to classify is $O((\sum_i |t_i|)^2)$, which is far better than the exponentially many full relation instances. There is also no ambiguity when labeling pairs as positive or negative when constructing the training data. Finally, we can rely on previous work on classification for binary relation extraction to identify pairs of related entities.

To train a classifier to identify pairs of related entities, we must first create the set of all positive and negative pairs in the data. The positive instances are all pairs that occur together in a valid tuple. For the example sentence in Figure 1, these include the pairs (*John*, *CEO*), (*John*, *Inc. Corp.*), (*CEO*, *Inc. Corp.*), (*CEO*, *Biz. Corp.*), (*Jane*, *CEO*) and (*Jane*, *Biz. Corp.*). To gather negative instances, we extract all pairs that never occur together in a valid relation. From the same example these would be the pairs (*John*, *Biz. Corp.*) and (*Jane*, *Inc. Corp.*).

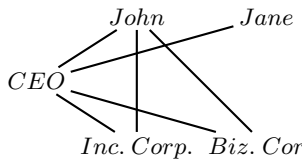
This leads to a large set of positive and negative binary relation instances. At this point we could employ any binary relation classifier and learn to identify new instances of related pairs of entities. We use a standard maximum entropy classifier (Berger et al., 1996) implemented as part of MALLET (McCallum, 2002). The model is trained using the features listed in Table 1.

This is a very simple binary classification model. No deep syntactic structure such as parse trees is used. All features are basically over the words separating two entities and their part-of-speech tags. Of course, it would be possible to use more syntactic information if available in a manner similar to that of Zelenko et al. (2003). However, the primary purpose of our experiments was not to create a better binary relation extractor, but to see if complex relations could be extracted through binary factorization followed by reconstruction. In Section 5.2 we present an empirical evaluation of the binary relation classifier.

Feature Set
entity type of e_1 and e_2
words in e_1 and e_2
word bigrams in e_1 and e_2
POS of e_1 and e_2
words between e_1 and e_2
word bigrams between e_1 and e_2
POS between e_1 and e_2
distance between e_1 and e_2
concatenations of above features

Table 1: Feature set for maximum entropy binary relation classifier. e_1 and e_2 are entities.

a. Relation graph G



b. Tuples from G
(*John*, *CEO*, $\perp$)
(*John*, $\perp$, *Inc. Corp.*)
(*John*, $\perp$, *Biz. Corp.*)
(*Jane*, *CEO*, $\perp$)
($\perp$, *CEO*, *Inc. Corp.*)
($\perp$, *CEO*, *Biz. Corp.*)
(*John*, *CEO*, *Inc. Corp.*)
(*John*, *CEO*, *Biz. Corp.*)

Figure 2: Example of a relation graph and tuples from all the cliques in the graph.

4.2 Reconstructing Complex Relations

4.2.1 Maximal Cliques

Having identified all pairs of related entities in the text, the next stage is to reconstruct the complex relations from these pairs. Let $G = (V, E)$ be an undirected graph where the vertices V are entity mentions in the text and the edges E represent binary relations between entities. We reconstruct the complex relation instances by finding maximal cliques in the graphs.

The simplest approach is to create the graph so that two entities in the graph have an edge if the binary classifier believes they are related. For example, consider the binary factorization in Figure 1 and imagine the classifier identified the following pairs as being related: (*John*, *CEO*), (*John*, *Inc. Corp.*), (*John*, *Biz. Corp.*), (*CEO*, *Inc. Corp.*), (*CEO*, *Biz. Corp.*) and (*Jane*, *CEO*). The resulting graph can be seen in Figure 2a.

Looking at this graph, one solution to construct-

ing complex relations would be to consider all the cliques in the graph that are consistent with the definition of the relation. This is equivalent to having the system return only relations in which the binary classifier believes that all of the entities involved are pairwise related. All the cliques in the example are shown in Figure 2b. We add $\perp$ fields to the tuples to be consistent with the relation definition.

This could lead to a set of overlapping cliques, for instance $(\textit{John}, \textit{CEO}, \textit{Inc. Corp.})$ and $(\textit{John}, \textit{CEO}, \perp)$. Instead of having the system return all cliques, our system just returns the maximal cliques, that is, those cliques that are not subsets of other cliques. Hence, for the example under consideration in Figure 2, the system would return the one correct relation, $(\textit{John}, \textit{CEO}, \textit{Inc. Corp.})$, and two incorrect relations, $(\textit{John}, \textit{CEO}, \textit{Biz. Corp.})$ and $(\textit{Jane}, \textit{CEO}, \perp)$. The second is incorrect since it does not specify the *company* slot of the relation even though that information is present in the text.

It is possible to find degenerate sentences in which perfect binary classification followed by maximal clique reconstruction will lead to errors. One such sentence is, *"John is C.E.O. and C.F.O. of Inc. Corp. and Biz. Corp. respectively and Jane vice-versa"*. However, we expect such sentences to be rare; in fact, they never occur in our data.

The real problem with this approach is that an arbitrary graph can have exponentially many cliques, negating any efficiency advantage over enumerating all n-tuples of entities. Fortunately, there are algorithms for finding all maximal cliques that are efficient in practice. We use the algorithm of Bron and Kerbosch (1973). This is a well known branch and bound algorithm that has been shown to empirically run linearly in the number of maximal cliques in the graph. In our experiments, this algorithm found all maximal cliques in a matter of seconds.

4.2.2 Probabilistic Cliques

The above approach has a major shortcoming in that it assumes the output of the binary classifier to be absolutely correct. For instance, the classifier may have thought with probability 0.49, 0.99 and 0.99 that the following pairs were related: $(\textit{Jane}, \textit{Biz. Corp.})$, $(\textit{CEO}, \textit{Biz. Corp.})$ and $(\textit{Jane}, \textit{CEO})$ respectively. The maximal clique method would not produce the tuple $(\textit{Jane}, \textit{CEO}, \textit{Biz. Corp.})$ since it never considers the edge between *Jane* and *Biz. Corp.* However, given the probability of the edges, we would almost certainly want this tuple returned.

What we would really like to model is a belief that *on average* a clique represents a valid relation instance. To do this we use the complete graph $G = (V, E)$ with edges between all pairs of entity mentions. We then assign weight $w(e)$ to edge e equal to the probability that the two entities in e are related, according to the classifier. We define the weight of a clique $w(C)$ as the mean weight of the edges in the clique. Since edge weights represent probabilities (or ratios), we use the geometric mean

$$w(C) = \left(\prod_{e \in E_C} w(e) \right)^{1/|E_C|}$$

We decide that a clique C represents a valid tuple if $w(C) \geq 0.5$. Hence, the system finds all maximal cliques as before, but considers only those where $w(C) \geq 0.5$, and it may select a non-maximal clique if the weight of all larger cliques falls below the threshold. The cutoff of 0.5 is not arbitrary, since it ensures that the average probability of a clique representing a relation instance is at least as large as the average probability of it not representing a relation instance. We ran experiments with varying levels of this threshold and found that, roughly, lower thresholds result in higher precision at the expense of recall since the system returns fewer but larger tuples. Optimum results were obtained for a cutoff of approximately 0.4, but we report results only for $w(C) \geq 0.5$.

The major problem with this approach is that there will always be exponentially many cliques since the graph is fully connected. However, in our experiments we pruned all edges that would force any containing clique C to have $w(C) < 0.5$. This typically made the graphs very sparse.

Another problem with this approach is the assumption that the binary relation classifier outputs probabilities. For maximum entropy and other probabilistic frameworks this is not an issue. However, many classifiers, such as SVMs, output scores or distances. It is possible to transform the scores from those models through a sigmoid to yield probabili-

ties, but there is no guarantee that those probability values will be well calibrated.

5 Experiments

5.1 Problem Description and Data

We test these methods on the task of extracting genomic variation events from biomedical text (McDonald et al., 2004b). Briefly, we define a variation event as an acquired genomic aberration: a specific, one-time alteration at the genomic level and described at the nucleic acid level, amino acid level or both. Each variation event is identified by the relationship between a *type* of variation, its *location*, and the corresponding state change from an *initial-state* to an *altered-state*. This can be formalized as the following complex schema

$$(\text{var-type}, \text{location}, \text{initial-state}, \text{altered-state})$$

A simple example is the sentence

"At codons 12 and 61, the occurrence of point mutations from G/A to T/G were observed"

which gives rise to the tuples

(*point mutation, codon 12, G, T*)
(*point mutation, codon 61, A, G*)

Our data set consists of 447 abstracts selected from MEDLINE as being relevant to populating a database with facts of the form: *gene X with variation event Y is associated with malignancy Z*. Abstracts were randomly chosen from a larger corpus identified as containing variation mentions pertaining to cancer.

The current data consists of 4691 sentences that have been annotated with 4773 entities and 1218 relations. Of the 1218 relations, 760 have two $\perp$ arguments, 283 have one $\perp$ argument, and 175 have no $\perp$ arguments. Thus, 38% of the relations tagged in this data cannot be handled using binary relation classification alone. In addition, 4% of the relations annotated in this data are non-sentential. Our system currently only produces sentential relations and is therefore bounded by a maximum recall of 96%. Finally, we use gold standard entities in our experiments. This way we can evaluate the performance of the relation extraction system isolated from any kind of pipelined entity extraction errors. Entities in this domain can be found with fairly high accuracy (McDonald et al., 2004b).

It is important to note that just the presence of two entity types does not entail a relation between them. In fact, 56% of entity pairs are not related, due either to explicit disqualification in the text (e.g. *"... the lack of G to T transversion ..."*) or ambiguities that arise from multiple entities of the same type.

5.2 Results

Because the data contains only 1218 examples of relations we performed 10-fold cross-validation tests for all results. We compared three systems:

- **MC:** Uses the maximum entropy binary classifier coupled with the maximal clique complex relation reconstructor.
- **PC:** Same as above, except it uses the probabilistic clique complex relation reconstructor.
- **NE:** A maximum entropy classifier that naively enumerates all possible relation instances as described in Section 4.1.

In training system **NE**, all incomplete but correct instances were marked as positive since we found this had the best performance. We used the same pairwise entity features in the binary classifier of the above two systems. However, we also added higher order versions of the pairwise features. For this system we only take maximal relations,that is, if $(\textit{John}, \textit{CEO}, \textit{Inc. Corp.})$ and $(\textit{John}, \perp, \textit{Inc. Corp.})$ are both labeled positive, the system would only return the former.

Table 2 contains the results of the maximum entropy binary relation classifier (used in systems **MC** and **PC**). The 1218 annotated complex relations produced 2577 unique binary pairs of related entities. We can see that the maximum entropy classifier performs reasonably well, although performance may be affected by the lack of rich syntactic features, which have been shown to help performance (Miller et al., 2000; Zelenko et al., 2003).

Table 3 compares the three systems on the real problem of extracting complex relations. An extracted complex relation is considered correct if and only if all the entities in the relation are correct. There is no partial credit. All training and clique finding algorithms took under 5 minutes for the entire data set. Naive enumeration took approximately 26 minutes to train.

ACT	PRD	COR
2577	2722	2101
Prec	**Rec**	**F-Meas**
0.7719	0.8153	0.7930

Table 2: Binary relation classification results for the maximum entropy classifier. ACT: actual number of related pairs, PRD: predicted number of related pairs and COR: correctly identified related pairs.

System	Prec	Rec	F-Meas
NE	0.4588	0.6995	0.5541
MC	0.5812	0.7315	0.6480
PC	0.6303	0.7726	0.6942

Table 3: Full relation classification results. For a relation to be classified correctly, all the entities in the relation must be correctly identified.

First we observe that the maximal clique method combined with maximum entropy (system **MC**) reduces the relative error rate over naively enumerating and classifying all instances (system **NE**) by 21%. This result is very positive. The system based on binary factorization not only is more efficient then naively enumerating all instances, but significantly outperforms it as well. The main reason naive enumeration does so poorly is that all correct but incomplete instances are marked as positive. Thus, even slight correlations between partially correct entities would be enough to classify an instance as correct, which results in relatively good recall but poor precision. We tried training only with correct and complete positive instances, but the result was a system that only returned few relations since negative instances overwhelmed the training set. With further tuning, it may be possible to improve the performance of this system. However, we use it only as a baseline and to demonstrate that binary factorization is a feasible and accurate method for extracting complex relations.

Furthermore, we see that using probabilistic cliques (system **PC**) provides another large improvement, a relative error reduction of 13% over using maximal cliques and 31% reduction over enumeration. Table 4 shows the breakdown of relations returned by type. There are three types of relations, 2-ary, 3-ary and 4-ary, each with 2, 1 and 0 $\perp$ arguments respectively, e.g.

System	2-ary	3-ary	4-ary
NE	760:1097:600	283:619:192	175:141:60
MC	760:1025:601	283:412:206	175:95:84
PC	760:870:590	283:429:223	175:194:128

Table 4: Breakdown of true positive relations by type that were returned by each system. Each cell contains three numbers, Actual:Predicted:Correct, which represents for each arity the actual, predicted and correct number of relations for each system.

$(\textit{point mutation}, \textit{codon 12}, \perp, \perp)$ is a 2-ary relation. Clearly the probabilistic clique method is much more likely to find larger non-binary relations, verifying the motivation that there are some low probability edges that can still contribute to larger cliques.

6 Conclusions and Future Work

We presented a method for complex relation extraction, the core of which was to factorize complex relations into sets of binary relations, learn to identify binary relations and then reconstruct the complex relations by finding maximal cliques in graphs that represent relations between pairs of entities. The primary advantage of this method is that it allows for the use of almost any binary relation classifier, which have been well studied and are often accurate. We showed that such a method can be successful with an empirical evaluation on a large set of biomedical data annotated with genomic variation relations. In fact, this approach is both significantly quicker and more accurate then enumerating and classifying all possible instances. We believe this work provides a good starting point for continued research in this area.

A distinction may be made between the factored system presented here and one that attempts to classify complex relations without factorization. This is related to the distinction between methods that learn local classifiers that are combined with global constraints after training and methods that incorporate the global constraints into the learning process. McCallum and Wellner (2003) showed that learning binary co-reference relations globally improves performance over learning relations in isolation. However, their model relied on the transitive property inherent in the co-reference relation. Our system can be seen as an instance of a local learner. Punyakanok

et al. (2004) argued that local learning actually outperforms global learning in cases when local decisions can easily be learnt by the classifier. Hence, it is reasonable to assume that our binary factorization method will perform well when binary relations can be learnt with high accuracy.

As for future work, there are many things that we plan to look at. The binary relation classifier we employ is quite simplistic and most likely can be improved by using features over a deeper representation of the data such as parse trees. Other more powerful binary classifiers should be tried such as those based on tree kernels (Zelenko et al., 2003). We also plan on running these algorithms on more data sets to test if the algorithms empirically generalize to different domains.

Perhaps the most interesting open problem is how to learn the complex reconstruction phase. One possibility is recent work on supervised clustering. Letting the edge probabilities in the graphs represent a distance in some space, it may be possible to learn how to cluster vertices into relational groups. However, since a vertex/entity can participate in one or more relation, any clustering algorithm would be required to produce non-disjoint clusters.

We mentioned earlier that the only restriction of our complex relation definition is that the arity of the relation must be known in advance. It turns out that the algorithms we described can actually handle dynamic arity relations. All that is required is to remove the constraint that maximal cliques must be consistent with the structure of the relation. This represents another advantage of binary factorization over enumeration, since it would be infeasible to enumerate all possible instances for dynamic arity relations.

Acknowledgments

The authors would like to thank Mark Liberman, Mark Mandel and Eric Pancoast for useful discussion, suggestions and technical support. This work was supported in part by NSF grant ITR 0205448.

References

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1).

D.M. Bikel, R. Schwartz, and R.M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning Journal Special Issue on Natural Language Learning*, 34(1/3):221–231.

C. Bron and J. Kerbosch. 1973. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577.

N. Collier, C. Nobata, and J. Tsujii. 2000. Extracting the names of genes and gene products with a hidden Markov model. In *Proc. COLING*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proc. NAACL*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Information Integration on the Web*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. ICML*.

A. K. McCallum. 2002. MALLET: A machine learning for language toolkit.

D.M. McDonald, H. Chen, H. Su, and B.B. Marshall. 2004a. Extracting gene pathway relations using a hybrid grammar: the Arizona Relation Parser. *Bioinformatics*, 20(18):3370–78.

R.T. McDonald, R.S. Winters, M. Mandel, Y. Jin, P.S. White, and F. Pereira. 2004b. An entity tagger for recognizing acquired genomic variations in cancer literature. *Bioinformatics*, 20(17):3249–3251.

S. Miller, H. Fox, L.A. Ramshaw, and R.M. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proc. NAACL*.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Learning via inference over structurally constrained output. In *Workshop on Learning Structured with Output, NIPS*.

Barbara Rosario and Marti A. Hearst. 2004. Classifying semantic relations in bioscience texts. In *ACL*.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. CoNLL*.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *JMLR*.