The Dissertation Committee for Razvan Constantin Bunescu
certifies that this is the approved version of the following dissertation:

Learning for Information Extraction: From Named Entity Recognition and Disambiguation To Relation Extraction

Committee:

Raymond J. Mooney, Supervisor

Inderjit S. Dhillon

Joydeep Ghosh

Andrew McCallum

Bruce Porter

Learning for Information Extraction: From Named Entity Recognition and Disambiguation To Relation Extraction

by

Razvan Constantin Bunescu, B.S.; M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2007

To my wonderful teachers

Acknowledgments

First and foremost, I want to thank my advisor Ray Mooney, whose guidance and active support have been essential in my development as a researcher. Our weekly discussions have been a constant source of insightful ideas, significantly shaping the main contributions of this thesis. His advice and words of encouragement have also helped me overcome less fruitful periods in my research, and have shown me the value of extracting useful lessons from any research experience. Ray's approach to research has fostered an environment in which the free exchange of ideas, collaboration as well as independent thinking were highly appreciated. I consider myself fortunate to have had the chance to grow and work in this environment.

I would like to thank the members of my thesis committee – Inderjit Dhillon, Joydeep Ghosh, Andrew McCallum, and Bruce Porter – for their insightful feedback in various stages of the thesis. I am also indebted to them for their excellent research and inspired teaching, which have significantly influenced my own research.

I am grateful to my academic siblings from the Machine Learning group for the countless interactions we had during the research meetings and for their entertaining company. Ruifang Ge, Lilyana Mihalkova, Sugato Basu, Misha Bilenko, Rohit Kate, Prem Melville, and Yuk Wah (John) Wong have often helped with paper reviews and practice talks, and have always been keen to find answers to my research questions. During the last two years, I have been very fortunate to work in the same office with two wonderful colleagues and friends – John and Lily. Our

discussions, marked by enthusiasm and empathy, have transformed office life into an unforgettable experience.

I have been very lucky to collaborate with a group of wonderful people: Ruifang Ge, Rohit Kate, and Yuk Wah Wong from the Machine Learning group; Arun Ramani and Edward Marcotte from the Institute for Cellular and Molecular Biology; Jeonghee Yi from IBM; and Marius Pasca from Google. In particular, I would like to thank Marius for facilitating me a summer internship at Google, and also for his help in securing a Google grant that supported the latest research described in my thesis.

I would also like to thank some of the members of the UTCS department staff for their help during the last six years. Gloria Ramirez and Katherine Utz have been incredibly resourceful and very prompt in helping me with numerous administrative issues. In preparing for conferences, or starting a new semester, I have also greatly benefited from the experienced help of Laurie Alvarez and Stacy Miller.

My friendship with Rares and Florina Neagu has had an inestimable influence in preserving a happy outlook on life, especially during my extended periods of solitude. I am grateful to them for the wonderful time that we have spent together, be it at home in Austin, or on the road to interesting destinations.

Lastly, I would like to thank my parents for believing in me and for encouraging me to pursue my dreams.

The research in this thesis was supported by the National Science Foundation under grants IIS-0325116 and EIA-0303609, and a gift from Google Inc.

Razvan Constantin Bunescu

The University of Texas at Austin

August 2007

Learning for Information Extraction: From Named Entity Recognition and Disambiguation To Relation Extraction

Publication No. _____

Razvan Constantin Bunescu, Ph.D.
The University of Texas at Austin, 2007

Supervisor: Raymond J. Mooney

Information Extraction, the task of locating textual mentions of specific types of entities and their relationships, aims at representing the information contained in text documents in a structured format that is more amenable to applications in data mining, question answering, or the semantic web. The goal of our research is to design information extraction models that obtain improved performance by exploiting types of evidence that have not been explored in previous approaches. Since designing an extraction system through introspection by a domain expert is a

laborious and time consuming process, the focus of this thesis will be on methods that automatically induce an extraction model by training on a dataset of manually labeled examples.

Named Entity Recognition is an information extraction task that is concerned with finding textual mentions of entities that belong to a predefined set of categories. We approach this task as a phrase classification problem, in which candidate phrases from the same document are collectively classified. Global correlations between candidate entities are captured in a model built using the expressive framework of Relational Markov Networks. Additionally, we propose a novel tractable approach to phrase classification for named entity recognition based on a special Junction Tree representation.

Classifying entity mentions into a predefined set of categories achieves only a partial disambiguation of the names. This is further refined in the task of Named Entity Disambiguation, where names need to be linked to their actual denotations. In our research, we use Wikipedia as a repository of named entities and propose a ranking approach to disambiguation that exploits learned correlations between words from the name context and categories from the Wikipedia taxonomy.

Relation Extraction refers to finding relevant relationships between entities mentioned in text documents. Our approaches to this information extraction task differ in the type and the amount of supervision required. We first propose two relation extraction methods that are trained on documents in which sentences are manually annotated for the required relationships. In the first method, the extraction patterns correspond to sequences of words and word classes anchored at two entity names occurring in the same sentence. These are used as implicit features in a generalized subsequence kernel, with weights computed through training of Support Vector Machines. In the second approach, the implicit extraction features are focused on the shortest path between the two entities in the word-word dependency

graph of the sentence. Finally, in a significant departure from previous learning approaches to relation extraction, we propose reducing the amount of required supervision to only a handful of pairs of entities known to exhibit or not exhibit the desired relationship. Each pair is associated with a bag of sentences extracted automatically from a very large corpus. We extend the subsequence kernel to handle this weaker form of supervision, and describe a method for weighting features in order to focus on those correlated with the target relation rather than with the individual entities. The resulting Multiple Instance Learning approach offers a competitive alternative to previous relation extraction methods, at a significantly reduced cost in human supervision.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

A vast amount of knowledge is contained nowadays in large repositories of unstructured or weakly structured text documents. Publicly available collections of documents range from very general and highly distributed (World Wide Web) to very specific and localized (MEDLINE). The utility of such document repositories would increase significantly if they could be used to reliably answer queries about relevant entities (e.g. people, companies), and their relationships (e.g. employee, owner, affiliation). Information Extraction (IE) (Grishman, 2003) is the task of identifying mentions of entities and their relationships in text documents, with the aim of structuring the text data in a form that is more amenable to database or data mining algorithms. Conceptually, IE subsumes three basic subtasks: *named entity recognition*, *named entity disambiguation*, and *relation extraction*.

Named entity recognition addresses the problem of locating textual mentions of predefined types of entities, where the entity categories can be very diverse, ranging from people and companies in business applications to cells and proteins in biomedical applications. For example, when given the sentence "One of the invited

talks at ACL in Ann Arbor this summer was by Michael Jordan", a name entity recognizer that is targeted to people and locations should identify the two named entities "Ann Arbor" of type LOCATION, and "Michael Jordan" of type PERSON.

Knowing that a particular occurrence of a name refers to a predefined type of entity is often insufficient to uniquely identify the corresponding entity. This is due to the fact that, in natural language, the same name can refer to more than one entity – for example "Michael Jordan" can refer to the Berkeley professor, or the basketball player. Named entity disambiguation is the task of identifying the entity that corresponds to a particular occurrence of a name in a text document. Disambiguating named entities is an important IE subtask, especially when information extracted from a particular document must be integrated with information about the same entity coming from other documents.

Assuming that the relevant named entities have been correctly identified, a further step in IE is to find predefined relationships between the extracted entities. Similarly to named entity recognition, relation extraction is usually targeted to a particular application domain. Consequently, the set of relevant relationships considered in relation extraction depends on the type of narrative – they can range from corporate acquisitions mentioned in newspaper corpora to protein interactions described in biomedical literature. For example, when given the sentence "One of the invited talks at ACL in Ann Arbor this summer was by Michael Jordan", a relation extraction system that is designed to identify relationships between people and locations should extract a LOCATEDAT relation between "Michael Jordan" and "Ann Arbor".

Information extraction can provide the structured input required by more sophisticated algorithms in data mining (Nahm & Mooney, 2000), question answering (Fleischman et al., 2003), or the semantic web (Cimiano et al., 2004). In particular, question answering can benefit from all three IE subtasks. For example, if

the question is "Which cities did Michael Jordan go to in 2000?", in compiling the corresponding answers (where an answer is a list of cities) one can apply the three IE subtasks as follows:

1. The answer type for the question above is CITY. Therefore, a named entity recognizer that is trained to extract locations of type CITY can be run over the entire corpus in order to identify all city mentions, thus significantly reducing the set of candidate answers.

2. A relation extraction system that is trained to identify LOCATEDAT relationships between people and locations can be applied on sentences where the name Michael Jordan and a city name co-occur, further reducing the set of candidate answers.

3. The name "Michael Jordan" can refer to more than one individual – for example, Wikipedia contains entries for six people with this name. Named entity disambiguation can be used to cluster the results returned by the relation extraction system, so that cities that belong to the same cluster correspond to the same "Michael Jordan" entity. A list of cities that is partitioned to reflect the various entities that have the same name "Michael Jordan" is going to be more informative for the user than an answer that contains a flat list of cities.

1.2 Thesis contributions

The goal of this thesis is to derive information extraction models with improved performance by exploiting types of evidence that have not been used in previous systems.

Since designing an IE system through introspection by a domain expert is a laborious and time consuming process, the focus of this thesis will be on methods

that automatically induce the extraction model by training on a dataset of manually labeled examples. Because every model is designed such that its parameters can be learned through training on supervised data, we also address efficiency related issues, such as the time complexity of the algorithms used for inference and training. The advantages of our proposed models are empirically validated through experimental evaluations in which the new method is compared against previously proposed methods. The contributions of this thesis are outlined below.

1.2.1 Named Entity Recognition

Previous approaches to this task have considered candidate named entities in isolation, thus ignoring potential correlations between them. We approach named entity recognition as a *collective* phrase classification problem and propose using the expressive framework of Relational Markov Networks (RMNs) (Taskar et al., 2002) in order to capture correlations between the labels of various types of candidate phrases. For example, long names and their corresponding acronyms tend to have the same entity type, and the same is true of repetitions of a candidate name inside the same document. These correlations are captured by an RMN graph that is associated with an entire document. The strength of the correlations is then estimated through training on a corpus of text documents manually labeled for named entities. During testing, the candidate entities in a given document are collectively classified by running an inference algorithm on the corresponding RMN graph. In general, exact inference in the resulting graph is intractable, therefore an approximate inference algorithm is used instead. However, we also show that efficient exact inference can be achieved for the basic phrase based classification approach to named entity recognition by exploiting the special structure of the problem in a suitable junction tree (Cowell et al., 1999) representation. Experimental results on biomedical and newspaper corpora show that a significant increase in performance is obtained

when the RMN framework is used for modeling mutual influences between multiple extractions from the same document.

1.2.2 Named Entity Disambiguation

Our approach to entity disambiguation is intrinsically dependent on a large dictionary that maps proper names to their possible named entity denotations - in our research the dictionary was compiled from Wikipedia (Remy, 2002), a large online encyclopedia. The method *detects* whether a proper name refers to a named entity contained in the dictionary, and *disambiguates* between multiple named entities that can be denoted by the same name. Both detection and disambiguation are formulated as a ranking problem, using a ranking function that takes as arguments a proper name and a named entity. The value computed by the ranking function is a measure of the similarity between the document context of the proper name and the text of the article corresponding to the argument entity. Standard similarity functions (e.g., TF-IDF cosine similarity (Baeza-Yates & Ribeiro-Neto, 1999)) are based on the assumption that both the context and the article have many relevant words in common. However, there are many cases in which the context and the article refer to the same concepts using different words and phrases (synonyms), and consequently measures like TF-IDF cosine similarity fail to capture the actual context-article similarity. We alleviate this problem by augmenting the similarity measure with correlations between context words and Wikipedia categories that contain the argument entity. The strength of the correlations is learned from a very large set of training examples that are automatically extracted from the hyperlinks contained in Wikipedia articles. Experimental results on the task of detecting and disambiguating named entities in Wikipedia show that the newly learned similarity function obtains significantly better disambiguation accuracy than the traditional TF-IDF cosine similarity.

1.2.3 Relation Extraction

In relation extraction, pairs of entities mentioned in the same sentence are classified as to whether they belong to a predefined set of relationships. We describe a set of approaches to learning relation extractors that differ especially in the type of supervision required.

Single Instance Learning

In the first, traditional approach, a machine learning algorithm is applied to a collection of documents that have been manually annotated for the relation of interest. The choice of the actual learning algorithm depends on the type of structural information available. For example, deep syntactic information provided by current parsers for new types of corpora such as biomedical text is seldom reliable, since most parsers have been trained on different types of narrative. If reliable syntactic information is lacking, sequences of words around and between the two entities can be used as alternative useful discriminators. Therefore, we first describe a generalization of subsequence kernels (Lodhi et al., 2002) for which the implicit features correspond to sequences of words and word classes anchored at the two entity names. We also propose an alternative method that is based on a dependency graph kernel, for domains where a reliable dependency analysis of the sentence is available. The new dependency kernel is based on the observation that most of the information relevant for relation extraction is contained in the shortest path between the two entities in the dependency graph of the sentence.

Both methods belong to the class of *Single Instance Learning (SIL)* approaches – each training example consists of a sentence containing the two entities, and a relationship label as assigned by a human annotator. The advantages of using the two new approaches are demonstrated through experimental evaluations in which they are compared against previously proposed relation extraction methods.

Multiple Instance Learning

In the second approach to relation extraction, the amount of supervision is reduced significantly to only a handful of pairs of entities known to exhibit or not exhibit a particular relationship. Bags of sentences containing the pairs are extracted from the web, and the subsequence kernel approach is extended to handle this weaker form of supervision. The new relation extraction method belongs to the class of *Multiple Instance Learning (MIL)* (Dietterich et al., 1997) approaches – a positive training example in MIL is a bag of sentences known to contain at least one positive sentence. For relation extraction, because the training bags are very few and relatively large, a straightforward application of any MIL algorithm is bound to be affected by two types of bias: words that are semantically correlated with either of the two relation arguments (Type I) or words that are correlated only to a specific relation instance (Type II) are given too much weight in the learned model, thus leading to extraction errors during testing. We address the first type of bias by integrating a set of appropriately induced word weights into the subsequence kernel. Experimental results demonstrate that the new approach is successful at reducing the undesired influence of the first type of bias, and that overall the new system can reliably extract relations from web documents.

1.3 Thesis Outline

Below is a summary of the remaining chapters in this thesis, with references to the relevant publications:

- **Chapter 2:** We present a collective approach to named entity recognition (Bunescu & Mooney, 2004), and a novel phrase classification model in which inference is both exact and tractable (Bunescu, 2004) .

- **Chapter 3:** This chapter describes a ranking approach to named entity disambiguation that exploits the rich structure of a large online encyclopedia (Bunescu & Pasca, 2006).

- **Chapter 4:** We first present two relation extraction systems which differ with respect to the depth of syntactic analysis required for the input sentences (Bunescu & Mooney, 2005b, 2005a). One of these methods is then extended to learn from a much weaker type of supervision (Bunescu & Mooney, 2007).

- **Chapter 5:** We discuss possible directions for future research based on the work presented in this thesis.

- **Chapter 6:** We conclude with a review of the main contributions of this thesis.

Chapter 2

Named Entity Recognition

2.1 Motivation

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. 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 (Verspoor et al., 2006), 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 2.1 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 2.1: Medline abstract with all protein names emphasized.

However, it is not always the case that repeated phrases have the same label. Figure 2.2 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 2.2, where both tokens 'nitric' and 'oxide' are first tagged as **O**utside, and then **I**nside a protein name. In Section 2.3 we show how to capture the correlations between the labels of repeated phrases so that all the exceptions above are taken into account.

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 34kDa protein, termed NOSIP (eNOS interaction protein), which avidly binds to the carboxyl terminal region of the eNOS oxygenase domain.

Figure 2.2: Medline abstract with all protein names emphasized.

The capitalization pattern of the name itself is another useful indicator, however it is not sufficient by itself, as similar patterns are also used for other types of biological entities such as cell types or amino acids (see Figure 2.3). Therefore, correlations between the labels of repeated phrases, or between acronyms and their long form can provide additional useful information. Our intuition is that a method that could use this kind of information would show an increase in performance, especially when doing extraction from biomedical literature, where phenomena like repetitions and acronyms are pervasive.

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 2.3: Medline abstract with all protein names emphasized.

2.2 Background

The task of automatically inducing named entity recognizers from training data has received a lot of attention in the past decade, consequently we observe a high diversity in the proposed approaches and the learning algorithms used therein. Most of the proposed systems can be classified into two basic types of approaches:

- **Phrase Classification**: Candidate phrases from a document are classified as to whether they are instances of a predefined set of entity types or not. This can be done by learning a multi-class classifier, where the number of classes is equal to the number of entity types plus one (for non-entity phrases). Usually, this process boils down to learning sets of extraction patterns, one set of patterns for each of the entity types.

- **Token Classification**: Word tokens in a document are sequentially classified as being **I**nside or **O**utside of a given named entity. This type of classification is commonly referred to as *tagging*: each word in a sequence of words is labeled with an **I**nside or **O**utside tag. Named entities are extracted by doing token classification and then assembling maximally contiguous sequences of **I**nside tokens.

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 Rapier and SRV belong to the category of phrase classification approaches, and so is our collective approach to named entity recognition (as described in Section 2.3). On the other hand, our method is firmly grounded in the expressive framework of graphical models, like most of the token classification approaches that will be described in the next section.

2.2.1 Token Classification Approaches

Hidden Markov Models

Hidden Markov Models (HMMs) (Rabiner, 1989) have been successfully used for speech recognition before becoming a model of choice for other natural language tasks such as part-of-speech (POS) tagging or named entity (NE) 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 to 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 2.4. In this and all subsequent figures, the X symbols are used to denote observations, while Y symbols refer to hidden variables (states or labels).

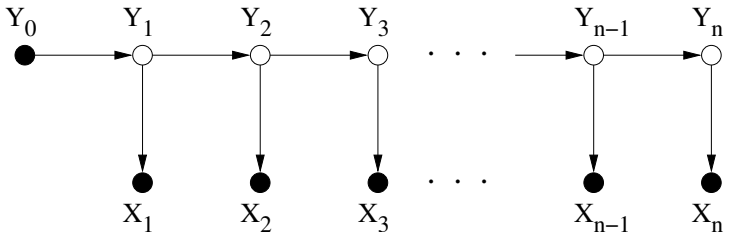


Figure 2.4: Unrolling an HMM as a directed graphical model.

One of the questions that an HMM inference algorithm is usually required to answer is to find the sequence of states that is most likely to have generated a given sequence of symbols. For example, in the case of NE recognition, each state corresponds to a named entity inside tag or the outside tag, whereas symbols correspond to words. Given a particular sentence, the named entities are defined by 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) (Rabiner, 1989). 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 et al., 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.

Numerous IE systems are 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 (Bikel et al., 1999; Freitag & McCallum, 1999).

Linear-Chain Conditional Random Fields

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. An HMM model is generative in the sense that it tries to model both the observation and hidden state sequences. However, in most applications 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 often unrealistic. The mismatch between model assumptions and data becomes even more pronounced if overlapping features based on observations such as word capitalization and part-of-speech are added to the model. Another inadequacy (McCallum et al., 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 of the model.

The Maximum Entropy principle has been widely used to create discriminative probabilistic models for natural language tasks (Berger et al., 1996). A maximum entropy approach to sequence tagging is able to accommodate a wide variety of overlapping features, based on binary tests on both the state and the observation context at the current position in the sequence. This type of model has been augmented by Ratnaparkhi (1996) to include features that relate the tags of two consecutive tokens. The new type of features 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. Such an approach is taken by a Maximum Entropy Markov Model (MEMM) (McCallum et al., 2000), which creates a maximum entropy model for each state in the model. For any 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. $p(s|s', o)$. Finding the most likely sequence of states in this context can be done efficiently using a Viterbi-like decoding algorithm.

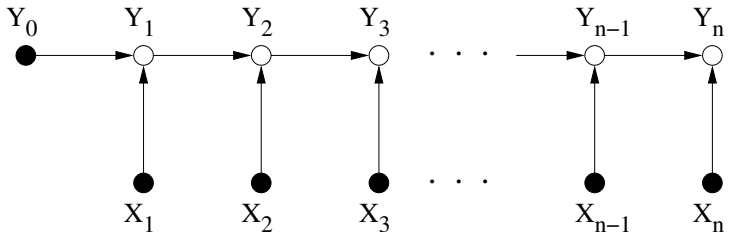


Figure 2.5: Unrolling an MEMM as a directed graphical model.

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 of 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 linear-chain Conditional Random Fields (CRFs) (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.

Inference in linear-chain CRFs can be done efficiently by accommodating the corresponding forward-backward or Viterbi algorithms used for HMMs (Rabiner,

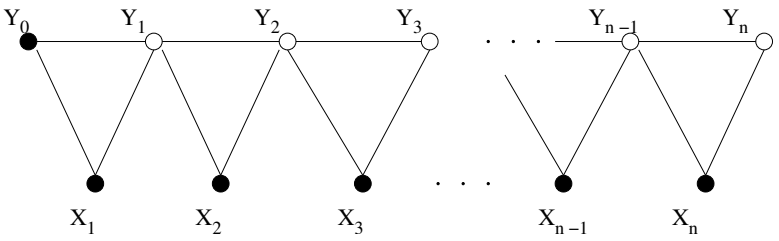


Figure 2.6: Unrolling a linear-chain CRF as an undirected graphical model.

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. Maximum Entropy models are a class of conditional models that subsume 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 affected by the label bias problem. This is particularly troublesome, as the problem does not occur with HMMs. The solution came in the form of linear-chain 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 section we give a more formal description of the framework underlying Conditional Random Fields.

2.2.2 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 (linear-chain CRFs, Boltzmann Machines).

Markov Random Fields are a type of undirected graphical model. 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 *(Li, 1995) V is said to be a Markov Random Field (MRF) with respect to G if for any vertex, its value depends only on its neighbors i.e. $\forall V_i \in V$, $P(V_i|V - V_i) = P(V_i|N(V_i))$.*

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 $\forall Y_i \in Y$, $P(Y_i|X, Y - Y_i) = P(Y_i|X, N(Y_i))$.*

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 *(Li, 1995) V is said to be a Gibbs Random Field (GRF) 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 that associates a positive number with each possible assignment of values to the clique vertices. Intuitively, the clique potential provides a quantitative measure for the compatibility between the values associated to vertices in a clique. The joint probability distribution over all vertices in the graph is obtained by taking a product over all 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 $P(Y|X) = \frac{1}{Z(X)} \prod_{c \in C(G)} \phi_c(X_c, Y_c)$, 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.

The following notational variants are often used in the graphical models literature:

- Markov Networks = Markov Random Fields;
- Conditional Random Fields = Conditional Markov Random Fields.

Relational Markov Networks

Relational Markov Networks (RMNs) (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. For example, the linear-chain CRF illustrated in Figure 2.6 can be specified by an RMN 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.

Relational Markov Networks are a general framework that allows for a compressed representation of undirected graphical models. RMNs can be seen as a convenient method for specifying general types of correlations between the attributes of entities in a particular domain. As such, they offer a suitable framework for our approach to "collective information extraction", as will be described in Section 2.3.

Factor Graphs

An alternative, useful representation for Markov random fields is provided by *factor graphs* (Kschischang et al., 2001). Factor graphs are bipartite graphs which express how a global function of many variables (the probability $P(Y|X)$ in Definition 4)

factors into a product of local functions (the potentials $\phi_c(X_c, Y_c)$ in Definition 4). 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 clique potential. Each potential node is then linked to all variable nodes from the associated clique. As an example, Figure 2.7 illustrates the factor graph corresponding to the linear-chain CRF from Figure 2.6.

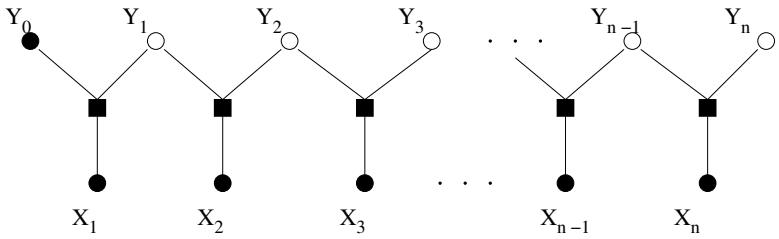


Figure 2.7: Representing a linear-chain CRF as a factor graph.

2.3 A Graphical Model for Collective NE Recognition

This section introduces our collective approach to named entity recognition as a phrase classification task. The expressive framework of Relational Markov Networks will be used in order to model correlations between the labels of candidate extractions from the same document. We start by defining a set of simple heuristics used to restrict the space of candidate entities, followed by a short description of the feature templates used to generate the actual features associated with each candidate extraction. We then show how the entity features can be modeled by *local clique templates* in the RMN framework, followed by the *global clique templates* used

to model the correlations between entity labels. The section ends with a description of the algorithms used for inference and learning.

2.3.1 Candidate Entities

Doing phrase classification requires a set of phrases to start with. Throughout this chapter, 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 by 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 part-of-speech tags (using the Penn tagset described in (Santorini, 1990)) 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 2.1 below shows the candidate entities generated by H1 and H2 on a fragment from a Medline abstract. Similarly, Table 2.2 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 2.1: 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 2.2: Candidate Extractions: CoNLL.

2.3.2 Entity Features

The set of features associated with each candidate is based on the feature templates introduced by 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 2.3, where we consider a generic candidate extraction as a sequence of $n+1$ words $w_0 w_1 ... w_n$.

Description	Feature Template
Head Word	$w_{(n)}$
Text	$w_{(0)}\_w_{(1)}\_\cdots\_w_{(n)}$
Short Type	$s_{(0)}\_s_{(1)}\_\cdots\_s_{(n)}$
Bigrams Left (4 bigrams)	$w_{(-1)}\_w_{(0)}$ $w_{(-1)}\_s_{(0)}$ $s_{(-1)}\_w_{(0)}$ $s_{(-1)}\_s_{(0)}$
Bigrams Right (4 bigrams)	$w_{(n)}\_w_{(n+1)}$ $w_{(n)}\_s_{(n+1)}$ $s_{(n)}\_w_{(n+1)}$ $s_{(n)}\_s_{(n+1)}$
Trigrams Left (8 trigrams)	$w_{(-2)}\_w_{(-1)}\_w_{(0)}$ ... $s_{(-2)}\_s_{(-1)}\_s_{(0)}$
Trigrams Right (8 trigrams)	$w_{(n)}\_w_{(n+1)}\_w_{(n+2)}$ ... $s_{(n)}\_s_{(n+1)}\_s_{(n+2)}$
Word, POS Left	$w_{(-1)}$ $t_{(-1)}$
Word, POS Right	$w_{(n+1)}$ $t_{(n+1)}$
Prefix (n+1 prefixes)	$s_{(0)}$ $s_{(0)}\_s_{(1)}$ ... $s_{(0)}\_s_{(1)}\_\cdots\_s_{(n+1)}$
Suffix (n+1 suffixes)	$s_{(n)}$ $s_{(n-1)}\_s_{(n)}$ ... $s_{(0)}\_s_{(1)}\_\cdots\_s_{(n+1)}$

Table 2.3: Feature 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 in the training data are filtered out.

2.3.3 The RMN Framework for NE 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 2.3.1). Each entity $e \in d.E$ is characterized by a predefined set of boolean attributes $e.F$ (Section 2.3.2), 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. Because the focus is on extracting protein names from Medline abstracts, in this section we consider only binary labels; however the model can also accommodate multiple entity types, as will be described later in Section 2.5.2.

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. $\forall s \in S_c$, $\phi_c(s) \geq 0$.

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 2.3.3 and 2.3.3 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 graphical 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:

$$P(d.Y|d.X) = \frac{1}{Z(d.X)} \prod_{c \in C} \prod_{G \in M_c(d.E)} \phi_C(G.X_c, G.Y_c) \tag{2.1}$$

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

$$Z(d.X) = \sum_{Y} \prod_{c \in C} \prod_{G \in M_c(d.E)} \phi_C(G.X_c, G.Y_c) \tag{2.2}$$

Local Clique Templates

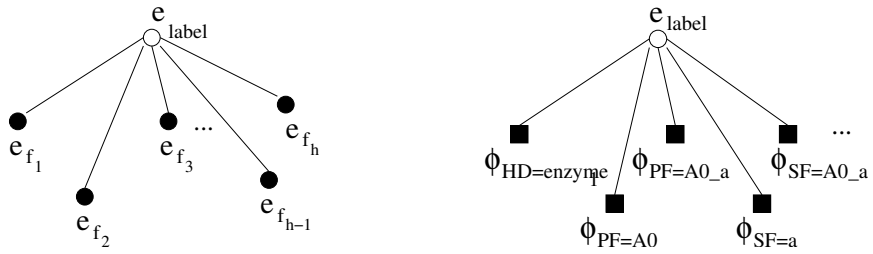
As described in the previous section, the role of local clique templates is to model correlations between an entity's observed features (see Table 2.3) and its label. If, after filtering, we are left with h distinct boolean features f_i, one way to model these correlations is to introduce h local (clique) 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 2.8 (a).

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).

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. However, the Gibbs distribution can be reduced by eliminating all potentials corresponding to zero-valued feature nodes. This is equivalent to a graphical model containing only nodes with a non-zero value. Consequently, the table associated with the local potential is reduced to contain only 2 values, specifying the compatibility between that feature and the two possible values for the entity label. As an example,



(a) RMN generated by local templates (b) Factor Graph for local templates.

Figure 2.8: Local templates.

Figure 2.8 (b) shows the factor graph that is generated around the entity label for 'HDAC1 enzyme' (with variable nodes figured as empty circles and potential nodes figured as black squares). Since we consider only features that have value 1, their influence on the entity labels is implicitly modeled through the potential nodes, hence the corresponding variable nodes can be safely ignored in the factor graph representation.

Global Clique Templates

Global clique templates facilitate the modeling of hypothesized influences between entities from the same document. They connect the label nodes of two or more entities, which, in the factor graph representation, translates into potential nodes connected to at least two label nodes. In our approach 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.

The Overlap Template

The definition of a *candidate extraction* from Section 2.3.1 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 2.4.

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

Table 2.4: 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 that the potential node will be connected to a variable number of entity label nodes. We will investigate this clique template in more detail later in Section 2.4.1.

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 2.3.3, 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 2.9 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.

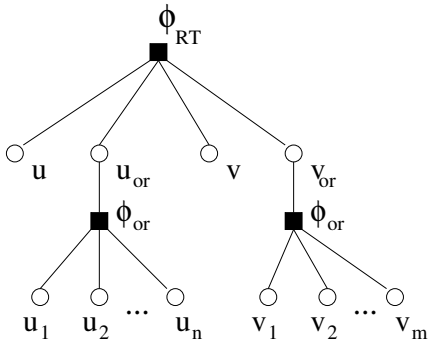


Figure 2.9: Repeat Factor Graph.

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'.

The Acronym Template

One approach to the acronym template would be to use an extant algorithm for identifying acronyms and their long forms in a document, and then define a potential function that would favor label configurations in which both the acronym and its definition have the same label. One such algorithm is described by Schwartz and Hearst (2003), achieving a precision of 96% at a recall rate of 82%. However, because this algorithm would miss a significant number of acronyms, we have decided to implement a softer version as follows: detect all situations in which a single word is enclosed between parentheses, such that the word length is at least 2 and it begins with a letter. Let v denote the corresponding entity. Let u_1, u_2, ..., u_n be all entities that end exactly before the open parenthesis. If this is a situation in which v is an acronym, then one of the entities u_i is its corresponding long form. Consequently, we use a logical OR template to introduce the auxiliary entity u_{OR}, and connect it to v's node label through an acronym potential ϕAT, as illustrated in Figure 2.10.

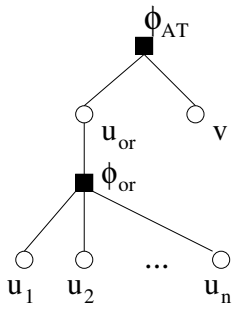


Figure 2.10: Acronym Factor Graph.

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 2.10. The candidate long forms are u_1 = 'antioxidant superoxide dismutase - 1', u_2 = 'superoxide dismutase - 1', and u_3 = 'dismutase - 1'.

2.3.4 Inference and Learning

So far, we have specified the theoretical apparatus needed to create the undirected graphical model corresponding to a given document: first, a set of candidate entities are selected based on one of the strong heuristics from Section 2.3.1; then, the actual graphical model is created by matching the clique templates introduced Section 2.3.3 against the selected set of candidate entities. There are two main problems that need to be solved once the undirected graphical model has been built:

1. **Inference:** Usually, two types of quantities are needed from the graphical 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:

$$d.Y^* = \underset{d.Y}{\operatorname{argmax}}\, P(d.Y|d.X) \tag{2.3}$$

where $P(d.Y|d.X)$ is defined as in Equation 2.1. A brute-force approach is excluded, since the number of possible label configurations is exponential in the num-

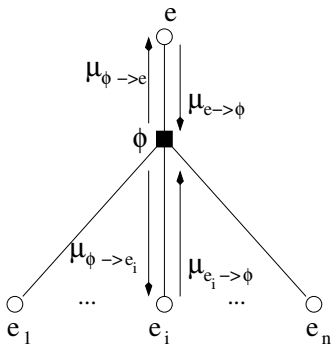
ber of candidate entities. The sum-product algorithm (Kschischang et al., 2001) is a message-passing algorithm that can be used for computing the marginal distribution over the label variables in factor graphs without cycles, and with a minor change (replacing the sum operator used for marginalization 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).

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 (and also for the second solution to overlap potential nodes), this step required optimization. Fortunately, due to the special form of the OR and overlap potentials (high degree of sparsity), and the normalization before each message-passing step, these special cases can be computed in linear-time. For example, the formulae for computing the OR messages for the sum-product algorithm are shown in Figure 2.11 (to avoid clutter, e and ϕ stand for e_{OR} and ϕ_{OR} respectively).

Learning the model parameters is defined as finding their maximum likelihood estimates. We use the following log-linear representation of potentials:

$$\phi_C(G.X_c, G.Y_c) = exp\{\mathbf{w_c}\mathbf{f_c}(G.X_c, G.Y_c)\} \tag{2.4}$$

Let $\mathbf{w}$ be the concatenated vector of all potential parameters $\mathbf{w_c}$. One approach to finding the maximum-likelihood solution for $\mathbf{w}$ is to use a gradient-based method, which requires computing the gradient of the log-likelihood with respect to potential



$$
\begin{aligned}
\mu_{\phi \to e}(0) &= \prod_{i=1}^{n} \mu_{e_i \to \phi}(0) \\
\mu_{\phi \to e}(1) &= 1 - \mu_{\phi \to e}(0) \\
\mu_{\phi \to e_i}(0) &= \mu_{\phi \to e}(1) + \\
&\quad \frac{\mu_{\phi \to e}(0)}{\mu_{e_i \to \phi}(0)} \left(\mu_{\phi \to e}(0) - \mu_{\phi \to e}(1)\right) \\
\mu_{\phi \to e_i}(1) &= \mu_{\phi \to e}(1)
\end{aligned}
$$

Figure 2.11: Messages in OR Factor Graph.

parameters $\mathbf{w_c}$. It can be shown that this gradient is equal to the difference between the empirical counts of $\mathbf{f_c}$ and their expectation under the current set of parameters $\mathbf{w}$.

$$\nabla L(w, D) = \sum_{d \in D} f_c(d.X, d.Y) - \sum_{d \in D} \sum_{d.Y'} f_c(d.X, d.Y') P_w(d.Y' | d.X) \tag{2.5}$$

The expectation in the second term is expensive to compute, since it requires summing over all possible configurations of candidate entity labels from a given document. To circumvent this complexity, we used Collins' voted perceptron approach (Collins, 2002), which can be seen as approximating the full expectation of $\mathbf{f_c}$ with the $\mathbf{f_c}$ counts for the most likely labeling under the current parameters $\mathbf{w}$.

$$\nabla L(w, D) \approx \sum_{d \in D} f_c(d.X, d.Y) - \sum_{d \in D} f_c(d.X, d.Y_w) \tag{2.6}$$

The Voted Perceptron algorithm is detailed in Table 2.5. At each step i in the algorithm, inference is performed using the current parameters w_i, so that we get the most likely labeling $d.Y_i$. The parameters are then updated based on the difference between the features counts computed on the ideal labeling $d.Y$ and those computed on the current most likely labeling $d.Y_i$. The final set of parameters is the average

Input: a set of documents D, number of epochs T, learning rate η.

set parameters $w_0 = 0$
set counter $i = 0$
for $t = 1...T$
  for every document $d \in D$
  $d.Y_i = \text{argmax}_{d.Y'} P_{w_i}(d.Y'|d.X)$
  $w_{i+1} = w_i + \eta * [f(d.X, d.Y) - f(d.X, d.Y_i)]$
  $i = i + 1$

Output: $w = \frac{1}{T\|D\|} \sum_i w_i$

Table 2.5: The Voted Perceptron Algorithm.

taken over the parameters at all steps i in the algorithm.

2.4 Exact & Tractable Inference for Phrase-Based NE Recognition

The problem of inference in general Markov Networks is known to be NP-hard (Cooper, 1990). Consequently, algorithms such as loopy belief propagation or generalized loopy belief propagation (Yedidia et al., 2000) are often used to do approximate inference. In this section we design an inference algorithm for local phrase based classification that is both exact and tractable.

By constraining the type of clique templates that are used to create the underlying graphical model, we can distinguish between the following two approaches to NE recognition:

- **[GT-RMN]** corresponds to an RMN that uses the *local templates* and all *global templates*. This Global Templates RMN stands for our approach to collective NE recognition introduced in Section 2.3.3.

- **[LT-RMN]** is the graphical model obtained by unrolling an RMN that uses only *local templates* and the *overlap template*. This Local Templates RMN

can be seen as the basic phrase classification approach to NE recognition – the only label correlations captured by the model are local, and they enforce the constraint that entity names should not overlap.

The sum-product algorithm is guaranteed to do exact inference for factor graphs without cycles. However, it happens very often that the factor graphs generated by our approach contain cycles, even in the LT-RMN model where only the overlap template is used. For example, if $w_1w_2w_3$ is a sequence of three nouns, the entire sequence, together with all its subsequences, will become candidate entities. We shall use the letter e to denote these candidate entities, as follows:

- unigram entities: $e_1 \leftarrow w_1$, $e_2 \leftarrow w_2$, $e_3 \leftarrow w_3$
- bigram entities: $e_{12} \leftarrow w_1w_2$, $e_{23} \leftarrow w_2w_3$
- trigram entities: $e_{123} \leftarrow w_1w_2w_3$

The corresponding set of overlapping pairs is $\{(e_1, e_{12}),\ (e_1, e_{123}),\ (e_2, e_{12}),\ (e_2, e_{23}),\ (e_2, e_{123}),\ (e_3, e_{23}),\ (e_3, e_{123}),\ (e_{12}, e_{23}),\ (e_{12}, e_{123}),\ (e_{23}, e_{123})\}$. The overlap template will create a two-node clique between the two nodes from each overlapping pair in the RMN factor graph, as illustrated in Figure 2.12. Of all cycles contained in this graph, we have emphasized using thick lines the cycle $e_1 - e_{12} - e_2 - e_{123} - e_1$, with the corresponding factor graph illustrated on the right.

Since the factor graphs in LT-RMN may contain cycles, the sum-product algorithm is not always guaranteed to do exact inference. Approximate inference may result in a significant number of extraction errors, which could explain why LT-RMN does not perform as well as linear-chain CRFs, for which the inference is exact (as shown by the comparative results from Section 2.5). In the remainder of this section we will show that exact inference can be done for the phrase based model LT-RMN in linear time, by using the junction-tree algorithm (Cowell et al., 1999) and by exploiting the sparsity of a different version of the overlap potential. Experimental

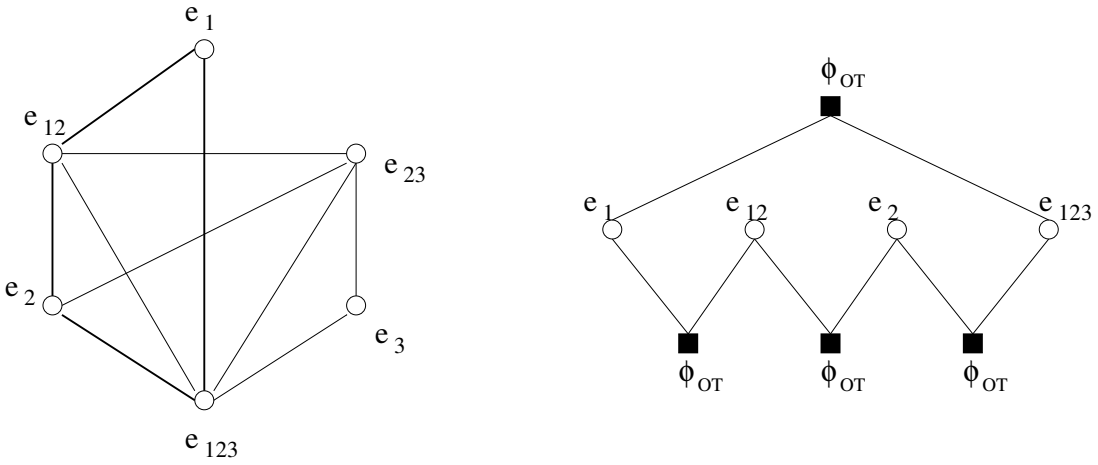


Figure 2.12: MRF and factor graph with cycles due to overlap clique potentials.

results will show that the new exact inference method leads to a significant increase in performance for the basic phrase based classification approach to NE recognition.

2.4.1 A Different Overlap Template

The original overlap template used in our approach from Section 2.3.3 creates an edge between any two overlapping entities. The constraint that entities should not overlap can also be enforced with a different type of overlap template, as follows. For a token position i in the document, let E_i be the set of candidate entities whose span includes that position. The overlap template is then defined so that, for each i, it connects the labels of all entities in E_i in a clique, with an associated potential that is non-zero only when at most one entity from E_i has label-value 1. Thus, if $|E_i| = n$, then the corresponding overlap potential can be specified as a table with 2^n entries, of which $n+1$ are set on 1, the rest being set to 0. This makes the potential table very sparse (a linear number of non-zero entries), a fact that will be used later in the inference algorithm. Notice that this version of the overlap template can match a variable number of entities, depending on the number of overlapping

entities at each token position.

Continuing with the same noun phrase example $w_1w_2w_3$, the sets of overlapping entities corresponding to the three token positions are:

1. $E_1 = \{e_1, e_{12}, e_{123}\}$
2. $E_2 = \{e_2, e_{12}, e_{23}, e_{123}\}$
3. $E_3 = \{e_3, e_{23}, e_{123}\}$

The overlap template creates three cliques, corresponding to the three sets E_1, E_2, and E_3. This results in the same graph as that from Figure 2.12, containing numerous cycles, which again means that the belief propagation algorithm (or the sum-product algorithm in the equivalent factor graph) is not guaranteed to result in exact inference.

2.4.2 Exact, Linear Time Inference

The junction tree algorithm (Cowell et al., 1999) is a generalization of the sum-product algorithm that can be used for exact inference in general graphs. It is based on the junction tree representation, which is a singly connected graph whose nodes are clusters of nodes from the original graph. The utility of the junction tree algorithm is however limited by the fact that its time complexity is exponential in the size of the largest cluster, which can get very large, especially when the original graph has cycles.

Definition 5 *(Cowell et al., 1999) $H = (H.V, H.E)$ is a cluster graph for $G = (G.V, G.E)$ if $H.V \subseteq 2^{G.V}$ e.g. any vertex in H is a cluster of vertices from G.* ■

Definition 6 *(Cowell et al., 1999) A cluster graph H is a* **junction tree** *for G if it has the following three properties:*

1. **singly connected:** *there is exactly one path between each pair of clusters.*

2. **covering:** *for each clique A of G there is some cluster C such that $A \subseteq C$.*

3. **running intersection:** *for each pair of clusters C and C' that contain a vertex $v \in G.V$, each cluster on the unique path between C and C' also contains v.* ■

In order to create a cluster graph having the running intersection property, one needs to *triangulate* the original graph. *Triangulation* refers to adding sufficient additional edges such that the graph contains no *chordless* cycles i.e. cycles of four or more distinct vertices without a short-cut. The cluster nodes in the junction tree are simply maximal cliques in the triangulated graph. It is usually the process of triangulation which leads to arbitrarily large cliques in the triangulated graph, which translates into arbitrarily large cluster nodes in the junction tree. Fortunately, as the next theorem asserts, the graphs created by the overlap template are already triangulated (e.g. *chordal*):

Theorem 2 *(Bunescu, 2004) Let $w_1 w_2 ... w_n$ be a word sequence, arbitrarily long (it may be the entire sequence of words from a document). Let E be an arbitrary set of candidate entities, and E_i the set of overlapping entities at position i, where $1 \leq i \leq n$. Let G be the graph created by the application of the overlap template e.g. the result of creating n cliques, one clique for each E_i, for all $1 \leq i \leq n$. Then the overlap graph G is a chordal graph.* ■

For example, it can be verified easily that the overlap graph in Figure 2.12 is a chordal graph. In Theorem 3 below, we create a particular cluster graph and show that it is a junction tree for the overlap graph by verifying directly the three properties from Definition 6. A similar argument can be used for proving Theorem 2.

Theorem 3 *(Bunescu, 2004) Keeping with the notation from Theorem 2, let H be a cluster graph for G, defined as follows:*

- $H.V = \{E_i | 1 \leq i \leq n\}$ *e.g. the sets of overlapping entities are vertices in the cluster graph.*
- $H.E = \{(E_i, E_{i+1}) | 1 \leq i \leq n-1\}$ *e.g. connect clusters corresponding to consecutive positions only (resulting in a list of clusters).*

Then H is a junction tree for G. ■

The result of applying this procedure on the overlap graph in Figure 2.12 is illustrated in Figure 2.13. Ellipses denote *cluster nodes*, while rectangles (*separator nodes*) are used to show the intersection between adjacent cluster nodes. It can be easily verified that this is a junction tree.

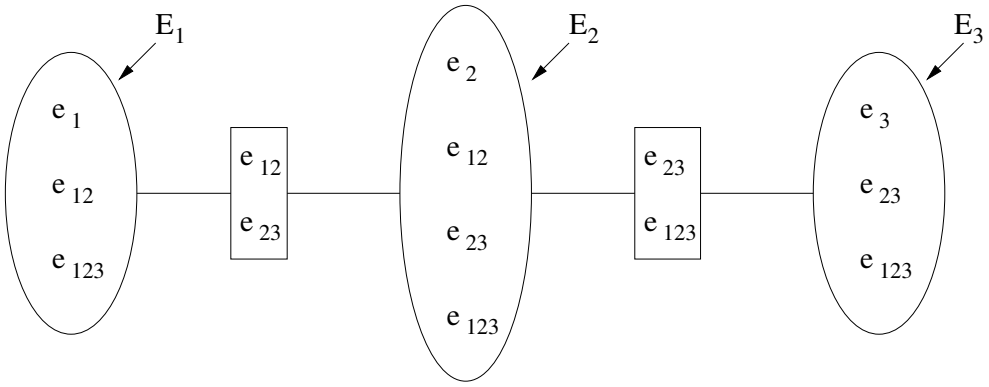


Figure 2.13: Sample junction tree.

Proof of Theorem 3. The first two properties from Definition 6 are directly verified by H. What is left to prove is the running intersection property. Let E_i and E_j be two cluster nodes. Assume without loss of generality that $i < j$. Let e be a vertex in the original graph G such that $e \in E_i$ and $e \in E_j$. Let $e.l$ and $e.r$ be the left and respectively right boundaries of entity e in the word sequence. Using the definition of the overlapping sets, $e \in E_i \Leftrightarrow e.l \leq i \leq e.r$, and $e \in E_j \Leftrightarrow e.l \leq j \leq e.r$. Let E_k be a cluster node on the path between E_i and E_j.

Because of the way in which H was created, k should be a position between i and j i.e. $i < k < j$. At this point, we have these three inequalities:

- $e.l \leq i \leq e.r$
- $e.l \leq j \leq e.r$
- $i < k < j$

Based on these inequalities, it results that $e.l \leq i < k < j \leq e.r$, therefore $e.l < k < e.r$. This implies that $e \in E_k$. As E_k was an arbitrary cluster on the path between E_i and E_j, this means that H has the running intersection property. Therefore, H is a junction tree for G. ■

Both Theorem 2 and Theorem 3 are important because they show that the size of the largest cluster in the junction tree (i.e. its *width*) is actually the size of the largest overlapping clique in the original graph. Because the inference algorithm using junction trees is in general exponential in this size, and because the size of the largest overlapping clique can be linear in the number of candidate entities, this means that exact inference using the generic junction tree algorithm is still exponential in the number of candidate entities. However, as Theorem 3 shows, for any overlapping graph, there exists a junction tree whose clusters are exactly the overlapping cliques. Because of the special form of the overlap clique potential (a sparse table, with only $n + 1$ non-zero entries, where n is the size of the clique), the messages sent between two adjacent *cluster nodes* in the junction tree can be computed in time linear in the size of the cluster. We therefore have an exact inference algorithm based on message propagation, where:

- The computation of any message takes time linear in the size of the adjacent cluster nodes.
- Assuming a two-phase propagation schedule (Jensen et al., 1990), the total number of messages is twice the number of cluster nodes.

- Assuming the length of any candidate entity is less than a maximum length (as is the norm in information extraction), the sum of all cluster sizes in the junction tree is linear in the total number of candidate entities.

Based on the three facts above, the overall time complexity of the message propagation algorithm in the junction tree structure from Theorem 3 is linear in the number of candidate entities.

2.4.3 Gradient-Based Learning

In Section 2.3.4 we have argued that since the gradient of the log-likelihood for the global GT-RMN model is generally expensive to compute, one could use instead the simpler voted perceptron algorithm in which only an approximate value of the gradient is calculated. In this section we derive a closed form formula of the gradient for the local LT-RMN model, which can be computed in time that is linear in the number of candidate entities. The exact value of the gradient is then provided to a gradient-based optimization algorithm in order to find a set of parameters that is globally optimal.

Because the overlap template potential is fixed, the only potential values that need to be learned are those used by local templates. Based on the same notation as in Section 2.3.4, we use the log-linear representation for a local template potential $\phi_c = exp(\mathbf{w_c}\mathbf{f_c})$. Being an exponential model, the gradient of the log-likelihood objective function $L(\mathbf{w}, D)$ with respect to the weight vector $\mathbf{w_c}$ is the difference between the observed and expected counts of the feature vector $\mathbf{f_c}$:

$$\begin{aligned} \nabla_c L(w, D) = \frac{\partial L(w, D)}{\partial w_c} &= \sum_{d \in D} f_c(d.X, d.Y) - \sum_{d \in D} \sum_{d.Y'} f_c(d.X, d.Y') P_w(d.Y' | d.X) \\ &= \sum_{d \in D} f_c(d.X, d.Y) - \sum_{d \in D} E_w[f_c(d)] \end{aligned} \tag{2.7}$$

While the first term in Equation 2.7 is easy to compute, the second term is usually expensive to compute in general graphical models. In the current setting however, we'll make use of the fact that all potentials involved are local potentials. Therefore, we can write:

$$f_c(d.X, d.Y) = \sum_{e \in d.E} f_c(e.X, e.Y) \tag{2.8}$$

Consequently, the expectation term in Equation 2.7 can be rewritten as follows:

$$\begin{aligned}
E_w[f_c(d)] &= \sum_{d.Y} \sum_{e \in d.E} f_c(e.X, e.Y) P_w(d.Y|d.X) && (2.9)\\
&= \sum_{e \in d.E} \sum_{d.Y} P_w(d.Y|d.X) f_c(e.X, e.Y) && (2.10)\\
&= \sum_{e \in d.E} \sum_{e.Y} \left(\sum_{d.Y \sim e.Y} P_w(d.Y|d.X) \right) f_c(e.X, e.Y) && (2.11)\\
&= \sum_{e \in d.E} \sum_{e.Y} P_w(e.Y|d.X) f_c(e.X, e.Y) && (2.12)
\end{aligned}$$

The expression $d.Y \sim e.Y$ above refers to all labelings of d consistent with a particular entity label $e.Y$. The term $P_w(e.Y|d.X)$ in the last equation is the marginal distribution for an entity label $e.Y$, which can be easily computed after running belief propagation in the junction tree, by selecting a cluster node containing e and marginalizing the cluster distribution over all other entities from the same cluster. Because the junction tree algorithm computes all clusters' marginal distributions at once, this means that computing the expectation term $E_w[f_c(d)]$ takes time linear in the number of candidate entities $d.E$. Based on the last equation, the final formula for the gradient is:

$$\nabla_c L(w, D) = \sum_{d \in D} \sum_{e \in d.E} \left(f_c(e.X, e.Y) - \sum_{e.Y'} P_w(e.Y'|d.X) f_c(e.X, e.Y') \right) \tag{2.13}$$

with a total computation time linear in the number of candidate entities. Maxi-

mum Likelihood (ML) estimation implicitly assumes a uniform prior distribution of the parameters $\mathbf{w}$. Alternatively, the parameters could be "smoothed" by using a Gaussian prior with zero mean and predefined variance σ^2, thus giving preference to more uniform models. The gradient for the maximum a posteriori (MAP) setting that uses a Gaussian prior is:

$$\nabla_c L(w, D) = \sum_{d \in D} \sum_{e \in d.E} \left(f_c(e.X, e.Y) - \sum_{e.Y'} P_w(e.Y'|d.X) f_c(e.X, e.Y') \right) - \frac{w_c}{\sigma^2} \tag{2.14}$$

Based on either formulation, any gradient-based method can be used for the ML or MAP estimation of the parameters. In our implementation we used L-BFGS, a limited-memory quasi-Newton method (Liu & Nocedal, 1989), which has shown very good performance elsewhere (Sha & Pereira, 2003).

In conclusion, we have introduced a discriminative model for information extraction based on phrase classification, in which exact inference is linear in the number of candidate phrases, and where both ML and MAP learning can be done efficiently. The overall approach is simple, and can be summarized as follows:

1. **Candidate Entities:** Based on the generic heuristic H1, or alternative domain-specific heuristics, create a set of candidate entities $d.E$, for each document in the corpus, $d \in D$.

2. **Junction Tree:** Assemble the set of candidate entities into cluster nodes E_i, one node for each token position i in the document. Each cluster E_i contains candidate entities that span over position i. Link cluster nodes corresponding to consecutive positions. The result is a list of cluster nodes, which by Theorem 3 is a junction tree for the original overlap graph. Depending on the set of candidate entities, some positions in the document may result in empty clusters, which split the junction tree into two or more smaller trees. This is also the case when the document is first split into sentences – because no

entity can belong to two different sentences, each sentence will have its own separate junction tree.

3. **Cluster Potentials:** Initialize each cluster potential with an overlap potential. Due to sparsity, if the cluster size is n, the cluster potential can be represented using only $n + 1$ parameters. Multiply all local template potentials for each entity into only one cluster potential. If there is more than one cluster containing the entity, choose one at random.

4. **Inference:** Run a message propagation algorithm on the resulting set of junction trees, using a two-phase propagation schedule.

5. **Learning:** Use a gradient based method in a ML or MAP setting, based on the gradient formula in Equation 2.13 or Equation 2.14 respectively.

Compared with CRFs, this has the additional benefit of allowing the incorporation of phrase-based features. Sarawagi and Cohen (2005) have introduced a conditional version for segmental semi-Markov models (Ge, 2002) to achieve a similar aim, showing that the phrase classification approach can lead to better performance vs. CRFs, especially when training data is small, because of a more natural use of phrase based features, such as similarities with existing dictionaries. Compared with their work, where sentences are modeled as Markov sequences of segments, our approach is more direct in modeling the extraction task as one of phrase classification. We explicitly model the entire set of candidate entities by including a node for each entity label in the graphical model.

2.5 Experimental Results

2.5.1 Systems

In order to evaluate the benefit of modeling correlations between candidate entities and the impact of the corresponding inference and learning algorithms, we compare the extraction performance of the following four systems:

- [**GT-RMN**] is our approach to collective NE recognition based on the RMN model that uses all *local* and *global clique templates*, as described in Section 2.3.3. Approximate inference is done using the sum-product algorithm for factor graphs, while learning is performed using the voted perceptron algorithm from Table 2.5 for 50 epochs, with a learning rate set at 0.01.

- [**LT-RMN**] is the graphical model obtained by unrolling an RMN that uses only *local templates* and the *overlap template*. The algorithms used for inference and learning for this phrase classification approach to NE recognition are the same as for GT-RMN.

- [**LT-JT**] corresponds to the phrase classification approach described in Section 2.4. In terms of local features and label constraints, this is equivalent with LT-RMN. However, the junction trees representation from Section 2.4.2 allows for an inference procedure that is both exact and tractable. For learning we use an implementation of the L-BFGS algorithm (Liu & Nocedal, 1989), coupled with the gradient computation from Section 2.4.3.

- [**CRF-TK**] is a CRF model for labeling token sequences based on the implementation of McCallum (2002). As in the maximum entropy approach of Bunescu et al. (2005), we use a comprehensive set of tags that distinguishes between five token types: outside tokens, beginning entity tokens, ending entity tokens, inside entity tokens and tokens from single word entities. The

token features are generated based on the features templates from Table 2.6, through which we try to emulate at the token level the entity level feature templates from Table 2.3. Like in the other three systems, features occurring less than three times in the training data are ignored.

Description	Feature Template
Current Word	$w_{(0)}$
Short Type	$s_{(0)}$
Bigrams Left (4 bigrams)	$w_{(-1)}$-$w_{(0)}$ $w_{(-1)}$-$s_{(0)}$ $s_{(-1)}$-$w_{(0)}$ $s_{(-1)}$-$s_{(0)}$
Bigrams Right (4 bigrams)	$w_{(0)}$-$w_{(+1)}$ $w_{(0)}$-$s_{(+1)}$ $s_{(0)}$-$w_{(+1)}$ $s_{(0)}$-$s_{(+1)}$
Trigrams Left (8 trigrams)	$w_{(-2)}$-$w_{(-1)}$-$w_{(0)}$ ... $s_{(-2)}$-$s_{(-1)}$-$s_{(0)}$
Trigrams Right (8 trigrams)	$w_{(0)}$-$w_{(+1)}$-$w_{(2)}$ ... $s_{(0)}$-$s_{(+1)}$-$s_{(2)}$
Word, POS Left	$w_{(-1)}$ $t_{(-1)}$
Word, POS Right	$w_{(+1)}$ $t_{(+1)}$

Table 2.6: CRF Feature Templates.

2.5.2 Datasets

The four systems above were tested on two datasets that have been hand-tagged for human protein names. The first dataset is Yapex (Franzen et al., 2002), 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 3,713 protein references. The second dataset is AIMed, which has been previously used for training the protein interaction extraction systems of Bunescu et al. (2005). 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 4,084 protein references in this dataset. All Medline abstracts are tokenized and then POS tagged using Brill's tagger (Brill, 1995).

In order to assess the impact of our collective approach to NE recognition on other types of narrative, we have also experimented with the CoNLL 2003 English corpus (Tjong Kim Sang & De Meulder, 2003) which contains four types of named entities: persons (PER), locations (LOC), organizations (ORG), and other (MISC). This leads to five possible label values (with a label-value of 0 indicating 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".

2.5.3 Results and Discussion

We run 10-fold cross-validation experiments, using the same folds for all systems, and pool the extraction results from all ten folds. Each extracted name in the test data is 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 for the protein recognition task are shown in Tables 2.7 and 2.8 which give average precision (P), recall (R), and F-measure (F).

$$P = \frac{\#\text{correct extractions}}{\#\text{extractions}}, \quad R = \frac{\#\text{correct extractions}}{\#\text{annotated extractions}}, \quad F = \frac{2P \times R}{P+R}.$$

Yapex			
Method	Precision	Recall	F-measure
GT-RMN	69.71	65.76	67.68
LT-RMN	70.79	53.81	61.14
LT-JT	72.08	57.46	63.95
CRF-TK	72.18	57.47	63.99

Table 2.7: Extraction Performance on Yapex.

The results show that, in terms of recall and F-measure, the use of global templates for modeling influences between candidate entities from the same document in GT-RMN significantly improves extraction performance over the local approach in LT-RMN (a one-tailed paired t-test for statistical significance results in a p value less than 0.01 on both datasets). In GT-RMN, a candidate phrase may be extracted, despite the fact that it appears in a non-informative context, if the same phrase is also repeated in a context that is very indicative of a protein name. It is therefore the ability to capture mutual influences between candidate extractions that allows GT-RMN to obtain a significant increase in recall. There is also a small improvement over CRF-TK, with the results being statistically significant only for the Yapex dataset, corresponding to a p value of 0.02.

AIMed			
Method	Precision	Recall	F-measure
GT-RMN	82.79	80.04	81.39
LT-RMN	81.33	72.79	76.82
LT-JT	81.76	75.11	78.29
CRF-TK	82.89	74.12	78.27

Table 2.8: Extraction Performance on AIMed.

The experimental results also show that the junction-tree version LT-JT

of the local model obtains a slightly better performance than the LT-RMN model. This is the expected behavior, since the junction tree algorithm does exact inference, whereas the sum-product algorithm in factor graphs is only an approximate inference algorithm. As shown in Section 2.4.2, exact inference can be done in time linear in the number of candidate entities by exploiting the sparsity of the overlap potentials. In terms of actual running time, on average, the exact inference implementation was around two times faster than the approximate inference algorithm. We have also run experiments with a version of LT-JT in which the parameters were learned using the voted perceptron algorithm. The results were very similar to those obtained using gradient-based learning, which means that exact inference is the actual cause for the observed increase in performance.

The overall results for the CoNLL dataset are presented in Table 2.9, with the global approach GT-RMN exhibiting improvement over its local version LT-RMN, albeit less pronounced than in the biomedical domain.

CoNLL			
Method	Precision	Recall	F-measure
GT-RMN	83.17	81.44	82.30
LT-RMN	82.15	78.13	80.09
LT-JT	82.14	79.80	80.95
CRF-TK	81.57	80.08	80.82

Table 2.9: Extraction Performance on CoNLL.

2.6 Related Work

Previous to our research, Chieu and Ng (2003) have exploited a set of global features in order to improve the accuracy of a Maximum-Entropy tagger; however, those 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.

Subsequent to the development of our method, there have been two alternative proposals for collective NE recognition: the skip-chain CRFs of Sutton and McCallum (2004), and the Gibbs sampling approach of Finkel et al. (2005). Both methods belong to the class of token based classification approaches, but differ with respect to the algorithm used for inference in the presence of long distance label correlations.

The skip-chain CRF of Sutton and McCallum (2004) is created by augmenting a linear-chain CRF with edges connecting similar (e.g. identical) tokens. Correspondingly, the probability distribution is modified with potential functions that depend on the context of every pair of similar tokens. Because the new edges introduce cycles in the underlying graphical model, exact inference becomes intractable, hence approximate inference is done using loopy belief propagation on the entire graph, guided by a tree based reparameterization (TRP) schedule for message passing (Wainwright et al., 2001).

Finkel et al. (2005) propose a significantly different approach that is centered on using Gibbs sampling (Geman & Geman, 1984) for inference. Long distance correlations are modeled through a multiplicative factor that penalizes a sequential structure based on the number of correlations that are violated, and the strength of the correlations as observed in the training data. Because Gibbs sampling proceeds by probabilistically changing only one state in the tag sequence, the probability distribution of the entire sequence can be updated dynamically in a very efficient way, even in the presence of long distance constraints.

While the two techniques are similar to our approach in the sense that they are both formulated inside the framework of undirected graphical models, each has its own advantages and disadvantages. Finkel's approach in particular looks very appealing, as it can enforce soft constraints that are based not only on the observed

features, but also on the labels. On the other hand, Gibbs sampling can be very time consuming, and it is unclear how good a performance it will obtain if applied to a phrase classification approach such as ours. We believe that doing NE recognition by phrase classification has the added benefit of allowing for phrase based features, which can have a significant impact in domains where entity dictionaries are available. Similarities between candidate entities and entries in the dictionary can be easily incorporated in a phrase classification approach, and can lead to improvements in extraction accuracy, especially when training data is scarce (Sarawagi & Cohen, 2005). Another advantage of phrase based classification is that, by relaxing the non-overlapping constraint, it could also allow for extractions that overlap. This can be very useful, since names often contain other names (e.g., "Interleukin-1" in "Interleukin-1 receptor", or "Japan" in "Bank of Japan"). Extracting all named entities, as opposed to extracting only the ones with the longest span, could provide a richer and more informative output.

The Semi-Markov Conditional Random Fields framework, which was proposed by Sarawagi and Cohen (2005), is another tractable approach to NE recognition modeled as phrase classification. We believe that the LT-JT method, which was first presented in (Bunescu, 2004), is significantly easier to implement. The inference in LT-JT is based on the traditional message passing algorithm, which is easy to understand and implement. Also, as shown in Section 2.4.3 on gradient-based learning, the expectation term from the gradient is simply a byproduct of inference (i.e., belief propagation) in the junction tree. In semi-Markov CRFs, inference is done using a complex semi-Markov analog of the Viterbi algorithm, while the gradient is computed with a significantly more complicated algorithm, completely separated from inference. LT-JT also explicitly creates an undirected graphical model that contains a node for each candidate phrase, making it easier to model long distance dependencies between phrases. While it is unclear how to accommodate such de-

pendencies in a semi-Markov CRF, they could be captured in the LT-JT model by adding an edge between the dependent phrases, an idea that we intend to pursue in future work.

2.7 Chapter Summary

In this chapter we have described two novel phrase based classification approaches to NE recognition. In the first model (GT-RMN), extraction is done by collectively classifying all candidate entities from the same document. We use the expressive framework of Relational Markov Networks in order to capture label correlations between repetitions, or between short and long forms of the same entity name. Experimental results demonstrate that the new global approach to NE recognition outperforms a local version that ignores label correlations between extractions. The new system also outperforms a token classification approach based on linear chain Conditional Random Fields, a state-of-the-art graphical model for sequence tagging. In the second model (LT-JT), we exploit the sparsity of the overlap potentials in a junction tree representation and obtain a new approach to NE recognition in which the phrase classification algorithm used for inference is both exact and tractable. We show that LT-JT obtains performance that is competitive to CRFs, and argue that the ability to easily incorporate phrase features makes LT-JT especially appropriate for domains in which named entity dictionaries are available.

Chapter 3

Named Entity Disambiguation

3.1 Motivation

As described in the previous chapter, in Named Entity Recognition one is concerned with finding textual mentions of predefined types of entities, such as people or companies in the news domain, or proteins and genes in the biomedical domain. In general, there is a many-to-many mapping between names and entities, which means that one cannot reason about entities based solely on the output of a NE recognizer. The many-to-many relationship between entities and names is caused by two different phenomena:

- **Polysemy:** The same name can refer to multiple entities.
- **Synonymy**: The same entity can have multiple names.

Polysemy is illustrated in the three sentences below, where the same name *John Williams* is used to refer to three different persons:

1. "*John Williams* and the Boston Pops conducted a summer Star Wars concert at Tanglewood."
2. "*John Williams* lost a Taipei death match against his brother, Axl Rotten."

3. "*John Williams* won a Victoria Cross for his actions at the battle of Rorke's Drift."

The entities denoted by the same name do not necessarily have to belong to the same category (e.g. people). For instance, *Venus* can refer to the second-closest planet to the Sun, or to the Roman goddess of love, as illustrated in the sentences below:

- "*Venus* is covered with an opaque layer of highly reflective clouds of carbon dioxide."
- "Paris decided in favor of *Venus* and gave her the golden apple."

In synonymy, the same entity may be mentioned using different names, as in the following sentence in which *Venus*, *Morning Star*, and *Evening Star* all refer to the same entity (i.e. the planet Venus).

- " *Venus* reaches its maximum brightness shortly before sunrise or shortly after sunset, for which reason it is often called the *Morning Star* or the *Evening Star*."

Synonymy is especially pervasive in the biomedical domain, where researchers do not always follow the standard nomenclature. For instance, it is not uncommon for a gene to have five or more alternative names, as illustrated below:

- **I1F5_HUMAN:** Interleukin 1 family member 5 | Interleukin-1 HY1 | Interleukin-1 delta | Interleukin-1 protein 1 | Interleukin-1 receptor antagonist homolog 1 | Interleukin-1-like protein 1 | IL-1 delta | IL-1 related protein 3 | IL-1F5 | IL-1HY1 | IL-1L1 | IL-1RP3 | IL-1ra homolog 1 | IL1F5 | IL1RP3 | FIL1 delta.

The Named Entity Disambiguation task is concerned with mapping occurrences of names in text documents to their corresponding denotations. A reliable

solution to the disambiguation problem would enable applications downstream on the NLP pipeline to reason about entities and thus provide results that more closely match the actual information need of the users.

Web Information Retrieval is one of the tasks that could benefit from NE disambiguation. Queries about named entities constitute a significant portion of popular web queries, according to search engine logs. When submitting queries such as *John Williams* or *Venus*, search engine users could also be presented with a compilation of facts and specific attributes about those named entities, rather than a set of best-matching web pages. One of the challenges in creating such an alternative search result page is the inherent ambiguity of the queries, as several instances of the same class (e.g., different people) or different classes (e.g., a planet, or a mythological entity) may share the same name in the query. The effectiveness of the search could be greatly enhanced if the search results were grouped together according to the corresponding sense, rather than presented as a flat, sense-mixed list of items (whether links to full-length documents, or extracted facts). As an added benefit, users would have easier access to a wider variety of results, whenever the top 10 or so results returned by the largest search engines happen to refer to only one particular (arguably the most popular) sense of the query (e.g., the *composer* in the case of *John Williams*), thus submerging or "hiding" documents that refer to other senses of the query.

Web Information Extraction is another NLP task where NE disambiguation could have a significant impact. In IE, like in many other natural language applications, significant performance gains can be achieved as a function of data size rather than algorithm complexity, as illustrated by the increasingly popular use of the web as a (very large) corpus (Dale, 2003). It seems therefore natural to try to exploit the web in order to also improve the performance of relation extraction, i.e. the discovery of useful relationships between named entities mentioned in text docu-

ments. A relation between two given entities can be asserted in multiple documents on the web. We believe that an IE system that exploited evidence from multiple text sources would see a significant gain in performance. However, if one wants to combine evidence from multiple web pages, then one needs again to solve the name disambiguation problem. Without solving it, a relation extraction system analyzing the sentence examples given above for *John Williams* could mistakenly consider the third as evidence that *John Williams* the composer fought at *Rorke's Drift*.

3.2 Background

3.2.1 Disambiguation vs. Discrimination

Named Entity Disambiguation associates names with entities that are predefined in an *external* repository. Building a comprehensive repository of entity definitions can be a very complex and laborious endeavor. For example, in the more general problem of Word Sense Disambiguation (WSD), a traditional source of sense definitions is WordNet (Fellbaum, 1998) – an "electronic lexical database" that has taken many person years to develop.

A slightly different task is that of Named Entity Discrimination, in which multiple occurrences of a name are clustered into classes by detecting whether two occurrences refer to the same entity. Essential to the distinction between disambiguation and discrimination is that the sense definitions be *external*. Without this condition, one could see disambiguation as discrimination followed by an extra step in which the sense associated with a cluster can be defined either by choosing one "representative" context from the cluster, or as the sum of all contexts in the cluster. When extended to general noun phrases, the discrimination problem is also known as coreference resolution.

Discrimination may be perceived as being easier than disambiguation because

we only need to determine if two names refer to the same entity and not what the actual entity is. Discrimination is clearly less demanding in terms of resources, as it does not require an external source of knowledge. However, both general and domain-specific sense repositories are already freely available, and most of them contain very useful types of rich structural information (e.g. taxonomies, hyperlink structures). As will be described later in Section 3.2.2, a disambiguation model can be trained to exploit the structural information contained in these repositories in order to improve the disambiguation, and implicitly the discrimination, accuracy.

The external sources used for sense definitions in named entity disambiguation can be very diverse; for instance, in the related task of WSD, people have used dictionaries, thesauri, bilingual corpora or hand-labeled training sets (Schutze, 1998). For the specific task of disambiguating person names, an interesting approach to entity definitions was taken by Bekkerman and McCallum (2005). In their approach, a person entity is implicitly defined by that person's social network. By design, our NE disambiguation method works with any type of named entities. The disambiguation algorithm is based on an entity dictionary derived from Wikipedia, a free online encyclopedia written collaboratively by volunteers. In the next section we give an overview of the Wikipedia structures that are most relevant to our approach to named entity disambiguation.

3.2.2 Wikipedia

Wikipedia is a free online encyclopedia written collaboratively by volunteers, using a wiki software that allows almost anyone to add and change articles. It is a multilingual resource – there are about 200 language editions with varying levels of coverage. Wikipedia is a very dynamic and quickly growing resource – articles about newsworthy events are often added within days of their occurrence. As an example, the September 2005 version contains 751,666 articles, around 180,000 more articles

than four months earlier. The work in this thesis is based on the English version from May 2005, which contains 577,860 articles.

Each article in Wikipedia is uniquely identified by its title – a sequence of words, with the first word always capitalized. Typically, the title is the most common name for the entity described in the article. When the name is ambiguous, it is further qualified with a parenthetical expression. For instance, the article on *John Williams* the composer has the title *John Williams (composer)*. Because each article describes a specific entity or concept, we consider the article to be the definition for that entity.

In general, there is a many-to-many correspondence between names and entities. This relation is captured in Wikipedia through *redirect* and *disambiguation* pages, as described in the next two sections.

Redirect Pages

Synonymy is explicitly modeled in Wikipedia through redirect pages. A *redirect page* exists for each alternative name that can be used to refer to an entity in Wikipedia. The name is transformed (using underscores for spaces) into a title whose article contains a redirect link to the actual article for that entity. For example, *John Towner Williams* is the full name of the composer *John Williams*. It is therefore an alternative name for the composer, and consequently the article with the title *John Towner Williams* is just a pointer to the article for *John Williams (composer)*. An example entry with a considerably higher number of redirect pages is *United States*. Its redirect pages correspond to acronyms (*U.S.A.*, *U.S.*, *USA*, *US*), Spanish translations (*Los Estados Unidos*, *Estados Unidos*), misspellings (*Untied States*) or synonyms (*Yankee land*).

Title	Redirect	Disambig
John Williams (composer)	John Towner Williams	John Williams
John Williams (wrestler)	Ian Rotten	John Williams
John Williams (VC)	*none*	John Williams
Boston Pops Orchestra	Boston Pops, The Boston Pops Orchestra	Pops
United States	US, USA, ... United States of America	US, USA, United States
Venus (planet)	Planet Venus	Venus, Morning Star, Evening Star

Table 3.1: Examples of Wikipedia titles and their aliases.

Disambiguation Pages

Polysemy is modeled in Wikipedia through *disambiguation pages*, which are created only for ambiguous names, i.e. names that denote two or more entities in Wikipedia. For example, the disambiguation page for the name *John Williams* lists 22 associated entities. Therefore, besides the non-ambiguous names that come from redirect pages, additional aliases can be found by looking for all disambiguation pages that list a particular Wikipedia entity. In his philosophical article "On Sense and Reference", Frege (1892) gave a famous argument to show that sense and reference are distinct. In his example, the planet Venus may be referred to using the phrases "morning star" and "evening star". This example is nicely captured in Wikipedia by two disambiguation pages, *Morning Star* and *Evening Star*, both listing Venus as a potential referent.

Categories

Every article in Wikipedia is required to have at least one category. As shown in Table 3.2, *John Williams (composer)* is associated with a set of categories, among them *Star Wars music*, *Film score composers*, and *20th century classical composers*. Categories allow articles to be placed into one or more topics. The topics can be

further categorized by associating them with one or more parent categories. In Table 3.2 *Venus* is shown as both an article title and a category. As a category, it has one direct parent *Planets of the Solar System*, which in turn belongs to two more general categories, *Planets* and *Solar System*. Thus, categories form a directed acyclic graph, allowing multiple categorization schemes to co-exist simultaneously. There are in total 59,759 categories in Wikipedia.

Title	Categories
John Williams (composer)	Star Wars music, Film score composers, 20th century classical composers, ...
John Williams (wrestler)	Professional wrestlers, People living in Baltimore
John Williams (VC)	British Army soldiers, British Victoria Cross recipients
Boston Pops Orchestra	American orchestras, Massachusetts musicians
United States	North American countries, Republics, United States
Venus (planet)	Venus, *Planets*, *Solar System*, *Planets of the Solar System*, ...

Table 3.2: Examples of Wikipedia titles and their categories.

Hyperlinks

Articles in Wikipedia often contain mentions of entities that already have a corresponding article. When contributing authors mention an existing Wikipedia entity inside an article, they are required to link at least its first mention to the corresponding article, by using *links* or *piped links*. Both types of links are exemplified in the following wiki source code of a sentence from the article on Italy: *"The [[Vatican City|Vatican]] is now an independent enclave surrounded by [[Rome]]"*. The string from the second link (*"Rome"*) denotes the title of the referenced article. The same string is also used in the display version. If the author wants another string displayed (e.g., *"Vatican"* instead of *"Vatican City"*), then the alternative string is included in a piped link, after the title string. Consequently, the display string

for the aforementioned example is: *"The Vatican is now an independent enclave surrounded by Rome"*.

3.3 Using Wikipedia for NE Disambiguation

In this section we show how to process the Wikipedia structures enumerated above in order to obtain a dictionary that links names to their possible denotations, and a dataset of disambiguated name occurrences. This will be followed by a description of our ranking approach to named entity disambiguation, for which the disambiguation dataset will be used to provide training and testing data.

The main concepts introduced so far are summarized using the notation below:

- Let E denote the entire set of entities from Wikipedia.
- For any entity $e \in E$:
  - Let $e.name$ be the name obtained from the corresponding article title by removing any expression between parentheses.
  - Let $e.T$ be the text of the article.
  - Let $e.R$ be the set of all names that redirect to e.
  - Let $e.D$ be the set of names whose disambiguation pages contain a link to e.
  - Let $e.C$ be the set of categories to which e belongs (i.e. e's immediate categories and all their ancestors in the Wikipedia taxonomy).

3.3.1 The Named Entity Dictionary

We organize all named entities from Wikipedia into a dictionary structure D, where each entry $d \in D$ has two fields: the name $d.name$, and the set of entities $d.E$ that

can be denoted by the name $d.name$ in Wikipedia.

The first step in building the dictionary is to identify named entities, i.e. entities with a proper name title. Because every title in Wikipedia must begin with a capital letter, the decision whether a title is a proper name relies on the following sequence of heuristic steps:

1. If e's title is a multiword title, check the capitalization of all content words, i.e. words other than prepositions, determiners, conjunctions, relative pronouns or negations. If all content words are capitalized, then e is a named entity. Otherwise, go to step 2.

2. If e's title is a one word title that contains at least two capital letters, then e is a named entity. Otherwise, go to step 3.

3. Count how many times $e.name$ occurs in the text of the article, in positions other than at the beginning of sentences. If at least 75% of these occurrences are capitalized, then e is a named entity.

The above heuristic procedure extracts close to half a million named entities from Wikipedia. The second step constructs the actual dictionary D as follows:

- The set of names in D consists of all strings that may denote a named entity, i.e. if $e \in E$ is a named entity, then its title name $e.name$, its redirect names $e.R$, and its disambiguation names $e.D$ are all added as names in D.

- Each entry name $d.name \in D$ is mapped to $d.E$, the set of entities that $d.name$ may denote in Wikipedia. Consequently, a named entity e is included in $d.E$ if and only if $d.name = e.title$, $d.name \in e.R$, or $d.name \in e.D$.

3.3.2 The Disambiguation Dataset

Disambiguated name occurrences can be extracted from the hyperlinks contained inside Wikipedia articles. As described in Section 3.2.2, the Wikipedia guidelines

specify that at least the first occurrence of a name in an article must be linked to the corresponding Wikipedia article, if such an article exists. Therefore, if the display name is ambiguous, the link or the piped link is disambiguating it. We use the term named entity *query* to denote the occurrence in a Wikipedia article of a proper name for which we would like to retrieve the corresponding entity. Given a query q, let $q.name$ and $q.e$ denote the display name and the actual entity corresponding to that name, as specified in the hyperlink. We create a dataset Q of disambiguated named entity queries by including in it all ambiguous proper name queries i.e. all queries q such that there is a matching entry in the named entity dictionary $d \in D$ (i.e. $q.name = d.name$) for which $d.E \geq 2$. For all such queries, we also set the field $q.E$ to contain all entities that may be denoted by $q.name$ in Wikipedia (i.e. $q.E = d.E$).

The notation used for queries $q \in Q$ and their fields is summarized in the list below:

- $q.name$ is the name displayed in the article.
- $q.e$ is the actual entity referred in the article.
- $q.E$ is the set of entities that can be denoted by the name $q.name$ (e.g., $q.e \in q.E$).
- $q.T$ is a text window centered on the query name. The window size is set to 55, which is the value that was observed to give optimum performance in the related task of cross-document coreference (Gooi & Allan, 2004).

The application of the extraction procedure outlined above results in a dataset Q of $1,783,868$ disambiguated queries.

3.4 Ranking Models for Disambiguation

One solution to named entity disambiguation would be to model it as a classification problem. In this approach, a classifier would be created for every name, where the number of classes would be equal to the number of entities that correspond to that name in Wikipedia. Given a feature vector representation, this approach is equivalent to learning a separate set of weights for each named entity in Wikipedia, where the number of weights is usually proportional to the size of the vocabulary. This leads to memory requirements that are linear in the number of entities and the vocabulary size. Since the total number of named entities in Wikipedia is very large (virtually unbounded), the final space constraints might be hard to satisfy. Ideally, one would like the named entity disambiguation model to depend as little as possible on the total number of entities in Wikipedia.

An alternative approach is to cast disambiguation as a ranking problem. Using the notation introduced in the previous section, let q be an arbitrary named entity query q and $e \in q.E$ an entity that may correspond to $q.name$. Assuming that an appropriate scoring function $score(q, e)$ is available, finding the named entity corresponding to the query q can be defined as finding the entity that yields the highest score:

$$\hat{e} = \underset{e \in q.E}{\operatorname{argmax}}\, score(q, e) \tag{3.1}$$

If $\hat{e} = q.e$, then $\hat{e}$ represents the correct answer. Disambiguation methods will then differ based on the way they define the scoring function.

3.4.1 Ranking with Context-Article Similarity

One ranking function that we evaluated experimentally is based on the TF-IDF cosine similarity between the context of the query and the text of the entity's article:

$$score(q, e) = cos(q.T, e.T) = \frac{q.T}{\|q.T\|} \frac{e.T}{\|e.T\|}$$

The factors $q.T$ and $e.T$ are represented in the standard vector space model, where each component corresponds to a term in the vocabulary, and the term weight is the standard $tf \times idf$ score (Baeza-Yates & Ribeiro-Neto, 1999). The vocabulary V is created by reading all Wikipedia articles and recording, for each word stem w, its document frequency $df(w)$ in Wikipedia. Stopwords and words that are too frequent (20%) or too rare (0.1%)are discarded. A generic document d is then represented as a vector of length $|V|$, with a position for each vocabulary word. If $f(w)$ is the frequency of word w in document d, and N is the total number of Wikipedia articles, then the weight of word $w \in V$ in the $tf \times idf$ representation of d is:

$$d_w = f(w) \ln \frac{N}{df(w)} \tag{3.2}$$

A ranking approach based on cosine similarity has a very small memory footprint – the ranking model needs to store just a set of IDF word weights.

3.4.2 Ranking with Word-Category Correlations

An error analysis of the cosine-based ranking method reveals that, in many cases, the pair $\langle q, q.e \rangle$ fails to rank first, even though words from the query context unambiguously indicate $q.e$ as the actual denoted entity. In these cases, cue words from the context do not appear in $q.e$'s article due to two main reasons:

1. The article may be too short or incomplete.

2. Even though the article captures most of the relevant concepts expressed in the query context, it does this by employing synonymous words or phrases.

Latent Semantic Indexing (LSI) (Deerwester et al., 1990) can be seen as a possible solution to this problem. In LSI, both the query context and the article are represented in a space indexed by concepts, instead of words. The projection from words to concepts is created through a Singular Vector Decomposition (SVD) of a word-document matrix that captures word co-occurrence patterns in a separate collection of documents. The proponents of LSI argue that the dimensions corresponding to the dominant singular vectors correspond to concepts that generalize across words, capturing for example synonymy and polysemy relations.

However, in this thesis we take a different approach: we model the semantic relationship between words indirectly through correlations between words and categories from the Wikipedia taxonomy. The cosine similarity between q and e can be seen as an expression of the total degree of semantic correlation between words from the context of query q and a given named entity e. When the correlation is too low because the Wikipedia article for the named entity e does not contain all words that are relevant to e, it is worth considering the semantic correlation between context words and the categories to which e belongs. For illustration, consider the two queries for the name *John Williams* from Figure 3.1.

To avoid clutter, Figure 3.1 depicts only two entities with the name *John Williams* in Wikipedia: the composer and the wrestler. On top of each entity, the figure shows one of their Wikipedia categories (*Film score composers* and *Professional wrestlers* respectively), together with some of their ancestor categories in the Wikipedia taxonomy. The two query contexts are shown at the bottom of the figure. In the context on the left, words such as *conducted* and *concert* denote concepts that are highly correlated with the *Musicians*, *Composers* and *Film score composers* categories. On the other hand, their semantic correlation with other categories in

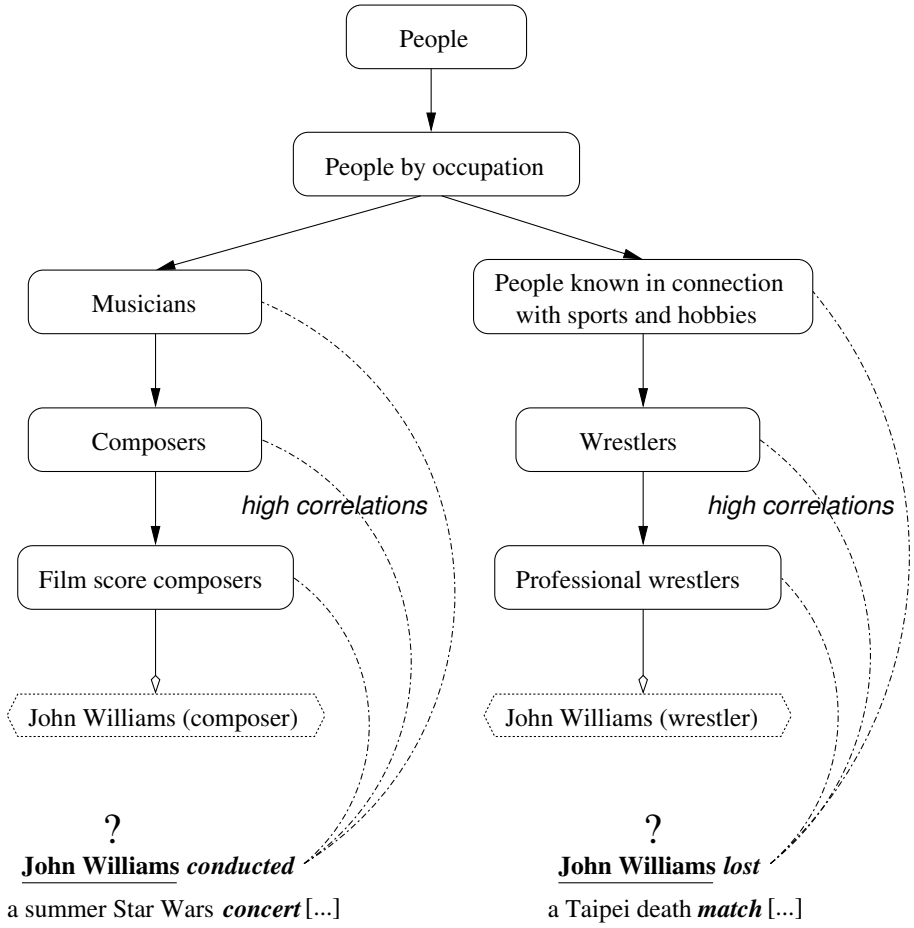


Figure 3.1: Word-Category correlations.

Figure 3.1 is considerably lower. Consequently, we want to design a disambiguation method that:

1. Learns the magnitude of semantic correlations between words and categories;
2. Exploits these correlations in a scoring function, together with the cosine similarity.

Our intuition is that, given the query context on the left, such a ranking function has a better chance of ranking the "composer" entity higher than the "wrestler"

entity, when compared with the simple cosine similarity baseline. We consider using a linear ranking function as follows:

$$\hat{e} = \underset{e \in q.E}{\operatorname{argmax}} \mathbf{w}\ \Phi(q, e) \tag{3.3}$$

The feature vector $\Phi(q, e)$ contains a dedicated feature ϕ_{cos} for cosine similarity, and $|V| \times |C|$ features $\phi_{w,c}$ corresponding to combinations of words w from the Wikipedia vocabulary V and categories c from the Wikipedia taxonomy C:

$$\begin{aligned} \Phi &= [\phi_{cos} | \Phi_{\mathbf{w},c}] \\ \phi_{cos}(q, e) &= cos(q.T, e.T) \\ \phi_{w,c}(q, e) &= \begin{cases} 1 & \text{if } w \in q.T \text{ and } c \in e.C, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{3.4}$$

One advantage of this ranking formulation is that, unlike in classification, the number of features is independent of the number of entities in Wikipedia. Figure 3.2 shows an example query for the name *John Williams*. The arcs correspond to word-category pairs $\langle w, c \rangle$, created from the context word "conducted" and some of the categories to which either of the two candidate entities belongs. Using the notation introduced earlier, the example from Figure 3.2 can be described as follows:

- Two named entities e_1 and e_2, corresponding to *John Williams (composer)* and *John Williams (wrestler)* respectively.
- One query q, characterized by the following fields:
  - $q.name =$ "$John\ Williams$";
  - $q.E = \{e_1, e_2\}$;
  - $q.e = e_1$;

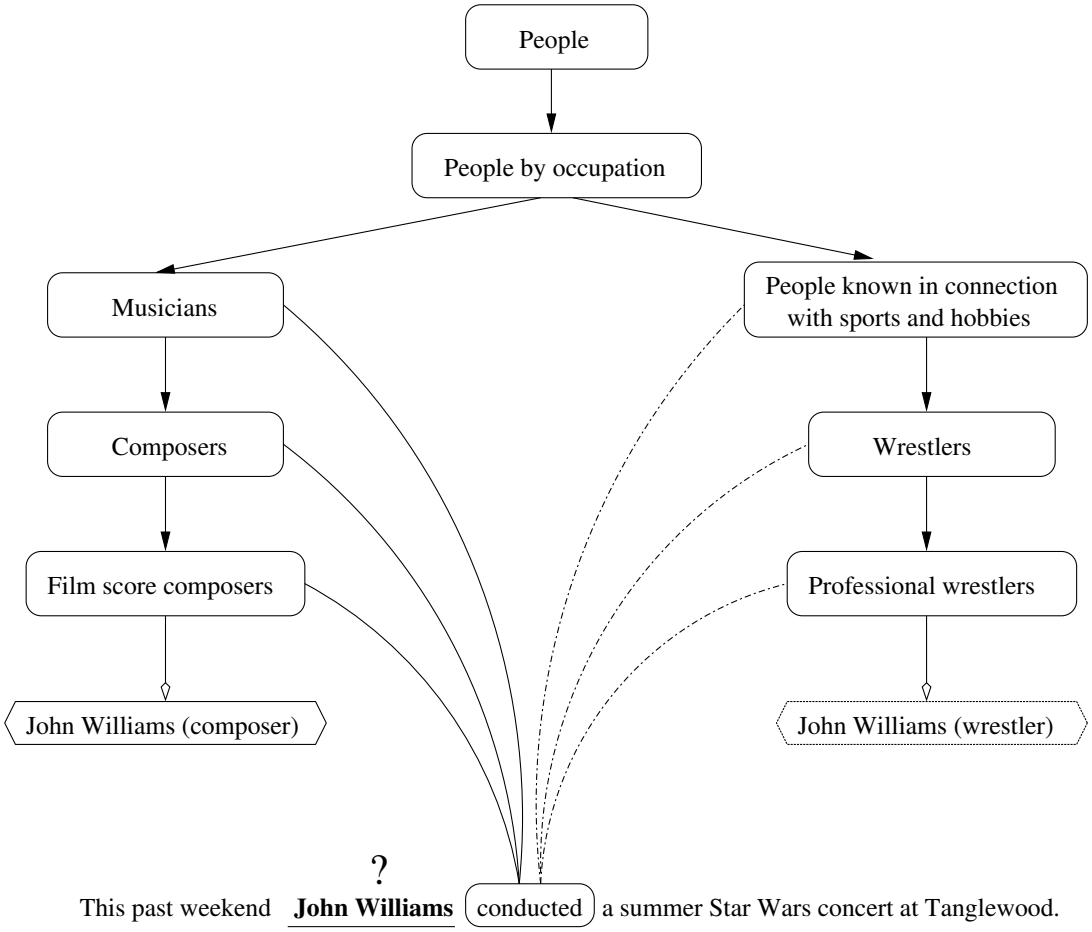


Figure 3.2: Query example.

– $q.T = \{$“$John\ Williams''$, “$past''$, “$week''$, “$conduct''$,
“$summer''$, “$Star\ Wars''$, “$concert''$, “$Tanglewood''$, ...$\}$.

The word-category features from the feature vector $\Phi(q, e)$ have binary values that depend on the text of the query q (i.e. $q.T$) and the categories (immediate and ancestor) to which the entity e belongs (i.e. the set $e.C$). Based on the data from Figure 3.2, we can create two feature vectors $\Phi(q, e_1)$ and $\Phi(q, e_2)$. Besides the cosine similarity feature ϕ_{cos}, they also contain a binary feature for every possible

pair $\langle w, c\rangle$ of words w from the Wikipedia vocabulary V and categories c from Wikipedia taxonomy C. As defined in Equation 3.4, the feature values depend on $q.T$ and $e.C$, with some examples shown below:

1. The first feature vector $\Phi(q, e_1)$:

   - $\phi_{w,c}(q, e_1) = 1$ for $\langle w, c\rangle \in \{$
     $\langle conducted, Composers\rangle, \langle conducted, Musicians\rangle,$
     $\langle conducted, People\rangle, ...,$
     $\langle concert, Composers\rangle, \langle concert, Musicians\rangle,$
     $\langle concert, People\rangle, ...\};$
   - $\phi_{w,c}(q, e_1) = 0$ for $\langle w, c\rangle \in \{$
     $\langle conducted, Wrestlers\rangle, \langle conducted, Professional\ Wrestlers\rangle, ...,$
     $\langle concert, Wrestlers\rangle, \langle concert, Professional\ Wrestlers\rangle, ...\};$

2. The second feature vector $\Phi(q, e_2)$:

   - $\phi_{w,c}(q, e_2) = 1$ for $\langle w, c\rangle \in \{$
     $\langle conducted, Wrestlers\rangle, \langle conducted, Professional\ Wrestlers\rangle, ...,$
     $\langle conducted, People\rangle, ...,$
     $\langle concert, Wrestlers\rangle, \langle concert, Professional\ Wrestlers\rangle, ...\},$
     $\langle concert, People\rangle, ...\};$
   - $\phi_{w,c}(q, e_2) = 0$ for $\langle w, c\rangle \in \{$
     $\langle conducted, Composers\rangle, \langle conducted, Musicians\rangle,$
     $\langle concert, Composers\rangle, \langle concert, Musicians\rangle, ...\};$

So far, the ranking approach to disambiguation implicitly assumes that Wikipedia contains all entities that may be denoted by names from the named entity dictionary. Taking for example the name *John Williams*, the present form of the ranking method assumes that, in any context, the referred entity is among the entities listed on the

disambiguation page in Wikipedia. In practice, there may be contexts where *John Williams* refers to an entity e_{out} that is not covered in Wikipedia, especially when e_{out} is not a popular entity. These *out-of-Wikipedia* entities are accommodated in the ranking approach to disambiguation as follows. A special entity e_{out} is introduced to denote any entity not covered by Wikipedia. Its attributes are set to null values as follows:

- The article text is empty: $e_{out}.T = \emptyset$;
- The set of categories is empty: $e_{out}.C = \emptyset$.

The ranking in Equation 3.1 can then be replaced with a new ranking procedure that returns the Wikipedia entity with the highest score, if this score is greater then a fix threshold τ, otherwise it returns e_{out}:

$$\begin{aligned} e_{max} &= \operatorname*{argmax}_{e \in q.E} score(q, e) \\ \hat{e} &= \begin{cases} e_{max} & \text{if } score(q, e_{max}) > \tau, \\ e_{out} & \text{otherwise.} \end{cases} \end{aligned} \tag{3.5}$$

However, the same behavior can also be implemented using the initial simple ranking formulation from Equation 3.3 if we redefine $q.E$ to include e_{out} (i.e. $q.E \leftarrow q.E \cup \{e_{out}\}$), and also add a new feature ϕ_{out} to the overall feature vector, as illustrated in Equation 3.6 below:

$$\begin{aligned} \Phi &= [\phi_{out}|\phi_{cos}|\Phi_{\mathbf{w},c}] \\ \phi_{out}(q, e) &= \mathbb{1}(e, e_{out}) \\ \phi_{cos}(q, e) &= cos(q.T, e.T) \\ \phi_{w,c}(q, e) &= \begin{cases} 1 & \text{if } w \in q.T \text{ and } c \in e.C, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{3.6}$$

The symbol $\mathbb{1}$ used in the formulation above stands for the identity function. The threshold τ from Equation 3.5 is equivalent to the weight learned for the feature ϕ_{out} in the new formulation from Equation 3.6. Thus, one advantage of the new feature vector representation is that the threshold is learned along with the weights for the other features.

The magnitude of each word-category correlation is captured by the weight vector $\mathbf{w}$, which can be learned by training on the disambiguated query dataset described in Section 3.3.2. We use the kernel version of the large-margin ranking approach introduced by Joachims (2002) which solves the optimization problem in Figure 3.3. The aim of this formulation is to find a weight vector $\mathbf{w}$ such that:

1. The correct entity $q.e$ has a higher score than the other candidate entities $e \in q.E - \{q.e\}$, i.e. the number of ranking constraints $\mathbf{w}\ \Phi(q, q.e) \geq \mathbf{w}\ \Phi(q, e)$ from the training data that are violated is minimized;

2. The ranking function $score(q, e) = \mathbf{w}\ \Phi(q, e)$ generalizes well beyond the training data.

$$
\begin{aligned}
&\text{minimize:} \\
&\quad V(w, \xi) = \tfrac{1}{2}\,||\mathbf{w}||^2 + C \sum \xi_{q,e} \\
&\text{subject to:} \\
&\quad \mathbf{w}\,(\Phi(q, q.e) - \Phi(q, e)) \geq 1 - \xi_{q,e} \\
&\quad \xi_{q,e} \geq 0 \\
&\forall q, \forall e \in q.E - \{q.e\}
\end{aligned}
$$

Figure 3.3: Optimization problem.

C is a parameter that allows trading-off margin size against training error. The number of linear ranking constraints is $\sum_q (|q.E| - 1)$. In the dual formulation, the learned $\mathbf{w}$ is a linear combination of the feature vectors $\Phi(q, e)$, which makes it possible to use kernels (Vapnik, 1998). Using a kernel approach means that the feature vectors $\Phi(q, e)$ (which can be very large) do not need to be created explicitly.

By definition, the kernel corresponds to the dot product between two feature vectors $\Phi(q, e)$ and $\Phi(q', e')$. In Equation 3.6, the features $\Phi_{\mathbf{w},c}(q, e)$ were defined to be nonzero only for words from the query context and categories associated with the entity e. This means that the dot product between two feature vectors $\Phi_{\mathbf{w},c}(q, e)$ and $\Phi_{\mathbf{w},c}(q', e')$ is equal to the product between the number of common words in the contexts of the two queries and the number of categories common to the two named entities. Consequently, the ranking kernel corresponding to the overall feature vector representation Φ is:

$$\begin{aligned} K\big(\langle q, e\rangle, \langle q', e'\rangle\big) &= \Phi(q, e) \cdot \Phi(q', e') \\ &= \mathbb{1}(e, e_{out}) \cdot \mathbb{1}(e', e_{out}) \\ &+ cos(q.T, e.T) \cdot cos(q'.T, e'.T) \\ &+ \big|q.T \cap q'.T\big| \cdot \big|e.C \cap e'.C\big| \end{aligned}$$

Because the product of the cosine terms can be very small compared to the other products in the kernel (usually around 10^{-8} for positive query-entity pairs), the word-category component of the kernel is normalized and the cosine term is multiplied with a constant $\alpha = 10^8$:

$$\begin{aligned} K\big(\langle q, e\rangle, \langle q', e'\rangle\big) &= \mathbb{1}(e, e_{out}) \cdot \mathbb{1}(e', e_{out}) \\ &+ \alpha \cdot cos(q.T, e.T) \cdot cos(q'.T, e'.T) \\ &+ \frac{|q.T \cap q'.T|}{\sqrt{|q.T| \cdot |q'.T|}} \cdot \frac{|e.C \cap e'.C|}{\sqrt{|e.C| \cdot |e'.C|}} \end{aligned}$$

3.5 Experimental Results

3.5.1 Methodology and Datasets

The taxonomy kernel was trained using the SVM^{light} package (Joachims, 1999). As described in Section 3.3.2, through its hyperlinks, Wikipedia provides a dataset of 1,783,868 disambiguated queries that can be used for training a named entity disambiguator. The apparently high number of queries actually corresponds to a moderate size dataset, given that the space of parameters includes one parameter for each word-category combination. However, due to time constraints, the taxonomy kernel was evaluated under the following four scenarios using just a subset of disambiguated queries:

1. [S_1] The working set of Wikipedia categories C_1 is restricted to only the 110 top level categories under *People by occupation*. The query dataset used for training and testing is reduced to contain only ambiguous queries $\langle q, e \rangle$ for which any potential matching entity e belongs to at least one of the 110 categories (i.e. $e.C \cap C_1 \neq \emptyset$). The set of negative matching entities e is also reduced to those that differ from the true answer $q.e$ in terms of their categories from C_1 (i.e. $e.C \cap C_1 \neq q.e.C \cap C_1$). In other words, this scenario addresses the task of disambiguating between entities with different top-level categories under *People by occupation*. Disambiguation is done using the feature representation from Equation 3.4.

2. [S_2] In a slight generalization of [S_1], the set of categories C_2 is restricted to all categories under *People by occupation*. Each category must have at least 200 articles to be retained,which results in a total of 540 categories out of the 8202 categories under *People by occupation*. The query dataset is generated as in the first scenario by replacing C_1 with C_2.

3. [S_3] This scenario is similar with [S_2], except that each category has to contain at least 20 articles to be retained, leading to 2847 out of 8202 categories.

4. [S_4] This scenario uses the same categories as [S_2] (i.e. $C_4 = C_2$). In order to make the task more realistic, all queries from the initial Wikipedia dataset are considered as follows. For each query q, out of all matching entities that do not have a category under *People by occupation*, one is randomly selected as an *out-of-Wikipedia* entity. Then, out of all queries for which the true answer is an *out-of-Wikipedia* entity, a subset is randomly selected such that, in the end, the number of queries with *out-of-Wikipedia* true answers is 10% of the total number of queries. In other words, the scenario assumes the task is to detect if a name denotes an entity belonging to the *People by occupation* taxonomy and, in the positive cases, to disambiguate between multiple entities under *People by occupation* that have the same name. Detection and Disambiguation are done jointly using the feature representation from Equation 3.6.

The dataset for each scenario is split into a training dataset and a testing dataset which are disjoint in terms of the query names used in their examples. For instance, if a query for the name *John Williams* is included in the training dataset, then all other queries with this name are reserved for learning (and consequently excluded from testing). Using a disjoint split provides a measure of how well the learned word-category correlations transfer to unseen entities. In future work we plan to investigate a complementary way of using the training data, in which the article of each named entity is augmented with the contexts from all queries for which this entity is the true answer. This alternative method has the potential of boosting the accuracy for both disambiguation methods when the training and testing datasets are not disjoint in terms of the proper names used in their queries.

Table 3.3 shows a number of relevant statistics for each scenario: #Cat represents the number of Wikipedia categories, #Constr is the number of rank-

ing constraints used during training. The training and testing datasets are also characterized in terms of the number of queries and query-answer pairs.

		Training Dataset			Test Dataset	
	# Cat.	Queries	Pairs $\langle q, e \rangle$	# Constr.	Queries	Pairs $\langle q, e \rangle$
S_1	110	12,288	39,880	27,592	48,661	147,165
S_2	540	17,970	55,452	37,482	70,468	235,290
S_3	2,847	21,185	64,560	43,375	75,190	261,723
S_4	540	38,726	102,553	63,827	80,386	191,227

Table 3.3: Scenario statistics.

In order to reduce the training time in S_4, the termination error criterion ϵ from SVM^{light} is changed from its default value of 0.001 to 0.01. Also, in the same scenario, the threshold τ for detecting *out-of-Wikipedia* entities when ranking with cosine similarity is set to the value that gives highest accuracy on training data.

3.5.2 Results and Discussion

Table 3.4 shows the results obtained for each scenario: #SV is the number of support vectors, TK(A) and Cos(A) are the accuracy of the Taxonomy Kernel and the Cosine similarity respectively. The Taxonomy Kernel significantly outperforms the Cosine similarity in the first three scenarios, confirming our intuition that correlations between words from the query context and categories from Wikipedia taxonomy provide useful information for disambiguating named entities. In the last scenario, which combines detection and disambiguation, the gain is not that substantial. Most queries in the corresponding dataset have only two possible answers, one of them an *out-of-Wikipedia* answer, and for these cases the cosine is already doing well at disambiguation. We conjecture that a more significant impact would be observed if the dataset queries were more ambiguous.

The high number of support vectors – half the number of query-answer pairs in training data – suggests that all scenarios can benefit from more training data.

Scenario	# S.V.	TK(A)	Cos(A)
S_1	19,693	**77.2%**	61.5%
S_2	29,148	**68.4%**	55.8 %
S_3	36,383	**68.0%**	55.4%
S_4	35,494	**84.8%**	82.3%

Table 3.4: Comparative evaluation.

One method for making this feasible is to use the weight vector w explicitly in a linear SVM. Because much of the computation time is spent on evaluating the decision function, using w explicitly may result in a significant speed-up. The dimensionality of w (by default $|V| \times |C|$) can be reduced significantly by considering only word-category pairs whose frequency in the training data is above a predefined threshold.

3.6 Related Work

Since comprehensive named entity repositories like Wikipedia have emerged only recently, there have been very few approaches to general named entity disambiguation. In general, published approaches focus on either of two types of named entities: geographical locations and people.

In the geographical domain, the different locations that can be denoted by the same name are extracted from *gazetteers* such as GNIS [1] for U.S. locations, or World Gazetteer [2] for non-U.S. locations. A gazetteer is usually organized as a taxonomy, in which a node is mapped to a unique place by hierarchically specifying its name and the names of all the regions encompassing it. A typical disambiguation method for geographical locations is that proposed by Li et al. (2002): first, candidate senses are associated with each location mentioned in a document by looking it up in the gazetteer; next simple word patterns are used to eliminate non-geographical mentions, or to further refine the location type and eliminate incompatible senses;

[1] http://geonames.usgs.gov
[2] http://www.world-gazetteer

then the *one-sense-per-discourse* is applied, followed by the application of a graph search algorithm that tries to enforce coherence between the senses assigned to all location names inside a document. Names that remain unresolved are assigned a default sense that is automatically derived from downloaded web pages. A similar method is also proposed by Amitay et al. (2004) – in their approach, coherence between multiple sense assignments is achieved by preferring senses (i.e. unique locations) that share the same encompassing region, as specified in the gazetteer taxonomy.

Bekkerman and McCallum (2005) propose a collective approach to disambiguating names of people known to be related in a social network. Each person name is first submitted as a query to a search engine, and then the first 100 pages are downloaded. Disambiguation is defined in this context as finding which of the 100 pages refer to the entity that participates in the input social network. All extracted pages are instantiated as nodes in an undirected graph, with edges connecting pages with non-disjoint link structure. In their first approach, the authors enforce coherence between sense assignments by requiring that the subgraph containing the relevant pages for all entities be maximally connected, based on the observation that people who are related in a social network tend to be mentioned in web pages with shared link structure.

Hassell et al. (2006) disambiguate researcher names in DBWorld [3] posts by exploiting relational information contained in an ontology derived from the DBLP (Ley, 2002) database. Attributes such as affiliations, topics of interests, or collaborators are extracted from the ontology and matched against the text surrounding a name occurrence. The results of the match are then combined in a linear scoring function that ranks all possible senses of that name.

Since training data is usually expensive to acquire, the methods described

[3] http://www.cs.wisc.edu/dbworld

above are all unsupervised. In contrast, our supervised approach to named entity disambiguation is trained on a dataset of disambiguated name queries that are automatically extracted from Wikipedia articles. The method is also designed to detect when a name denotes an entity that is not covered in the ontology – this is not the case with gazetteer based methods discussed above, which are implicitly based on a closed world assumption.

As described in Section 3.4.2, LSI can be used to give a better estimate of the similarity between the context of a name occurrence and the article of a candidate entity. Another possibility is to use an external semantic network for obtaining word-similarity information. For example, in the "semantic smoothing" method of Siolas and d'Alchè Buc (2000), the similarity between two words is computed as the inverse of the distance between the two words in the WordNet hierarchy. The TF-IDF document vectors are then smoothed through a linear transformation in which they are multiplied with the word-similarity matrix. Cristianini et al. (2001) describe how both LSI and semantic smoothing can be efficiently incorporated into a Latent Semantic Kernel (LSK). When used in conjunction with polynomial or Gaussian constructions, the LSK implicitly operates in spaces that are indexed by tuples of concepts. Both LSI and semantic smoothing could be used in combination with our taxonomy based approach in order to further improve disambiguation accuracy.

3.7 Chapter Summary

In this chapter, we have described a novel approach to NE disambiguation that exploits the rich structure of an online encyclopedia. We use the *redirect* and *disambiguation* pages in Wikipedia to create a comprehensive dictionary that maps proper names to their possible entity denotations. Disambiguation is modeled as a ranking problem, in which candidate entities are ordered based on a linear scoring function that exploits correlations between context words and categories in the Wikipedia

taxonomy. The magnitude of each word-category correlation is learned by training a ranking SVM kernel on a dataset of disambiguated name occurrences, automatically extracted from the named entity *hyperlinks* in Wikipedia articles. Large scale experiments show that the use of word-category correlations leads to a substantial improvement in accuracy over the context-article similarity baseline.

Chapter 4

Relation Extraction

4.1 Motivation

In Named Entity Recognition and Disambiguation, the aim is to find and disambiguate textual mentions of predefined types of entities. Often the user's information need is more complex, and it may refer to relations between entities, as illustrated in the three queries below:

1. "What *proteins* interact with **IL-1 receptor-associated kinase 1**?";
2. "Which *companies* were acquired by **Google**?";
3. "In which *city* was **Mozart** born?";

Relation Extraction (RE) is the task of finding tuples of entities for which there exists textual evidence that supports a predefined type of relationship. For example, a relation extraction system that is designed to extract protein interactions, company acquisitions, or people birthplaces should be able to label the following sentences as positive evidence for the tuples ⟨**IL-1 receptor-associated kinase 1**, **Pellino2**⟩, ⟨**Google**, **YouTube**⟩, and ⟨**Mozart**, **Salzburg**⟩ respectively:

1. "The phosphorylation of **Pellino2** by activated **IRAK1** and **IRAK4** could trigger and modulate the translocation of IRAKs from complex I to II";

2. "Search engine giant **Google** has bought video-sharing website **YouTube** in a controversial $1.6 billion deal";

3. "**Wolfgang Amadeus Mozart** was born to **Leopold** and **Anna Maria Pertl Mozart**, in the front room of Getreidegasse 9 in **Salzburg**".

The target relationship usually takes as arguments only entities with a predefined type, as emphasized in the example queries given earlier. Thus, for the three example queries, the relevant entities should be *proteins*, *companies*, or *people* and *cities*. Hence the utility of named entity recognition (Chapter 2), which can be used to provide the appropriate input to relation extraction.

Also, since users are most often interested in relations between entities rather than their textual mentions, the mentions need to be linked to their proper denotations, which could be entries in protein databases, or Wikipedia articles. In the examples given above, the relation extraction system must decide whether the name in the query is coreferential with the same name in the corresponding sentence. There are also two cases of synonymy that must be solved properly: **Wolfgang Amadeus Mozart** vs. **Mozart**, and **IL-1 receptor-associated kinase 1** vs. **IRAK1**. Hence also the utility of named entity disambiguation (Chapter 3), which can be used to solve named entity coreference and synonymy.

4.2 Background

As argued for named entity recognition in Chapter 2, manually developing an IE system can be a very time consuming, laborious process. The same holds for relation extraction, where a domain expert (e.g., a biologist if the focus is on protein interactions) needs to come up with an extraction model (e.g, a set of extraction

patterns) that is both accurate and comprehensive. In this chapter, we explore relation extraction models whose parameters are automatically learned from supervised data. One advantage of the supervised learning approaches is that they require significantly less human effort. However, the type of supervision provided to the corresponding learning algorithm can vary greatly in terms of the amount of human effort involved. The following two main distinctions have been made in the past in this respect:

- Single Instance Learning (SIL) corresponds to the traditional view of supervised learning in which a model is trained on a dataset of examples, each of which has been manually labeled as either positive or negative.
- Multiple Instance Learning (MIL) is a learning framework introduced by Dietterich et al. (1997) in which a model is trained on a dataset of bags of examples. A bag is labeled as positive if it contains at least one positive example, otherwise it is labeled as negative. Only the bag labels are available to the learning algorithm.

In the next two sections we describe in more detail the two types of supervision in the context of learning for relation extraction.

4.2.1 Single Instance Learning Supervision for RE

We restrict relation extraction to finding relationships between entities mentioned in the same sentence. The RE task is then equivalent to classifying sentences as to whether they assert a relationship between the entities mentioned therein. Because the relations investigated in this chapter are all binary, a sentence containing more than two entities will be used to create two or more examples. More exactly, if a sentence contains n entities ($n \geq 2$), it is replicated into $\binom{n}{2}$ sentences, each containing only two entities. If the two entities are known to be in the relationship,

then the replicated sentence is added to the set of corresponding positive sentences, otherwise it is added to the set of negative sentences. During testing, a sentence having n entities ($n \geq 2$) is again replicated into $\binom{n}{2}$ sentences in a similar way. In each sentence example, the two entity names are replaced with two generic tags $\langle e_1 \rangle$ and $\langle e_2 \rangle$, so that the learned relation extraction model is independent of the actual names of the arguments.

Below we enumerate the two positive and one negative sentence examples generated from the protein interaction sentence mentioned earlier in Section 4.1:

1. (+) "The phosphorylation of $\langle e_1 \rangle$ by activated $\langle e_2 \rangle$ and IRAK4 could trigger and modulate the translocation of IRAKs from complex I to II";

2. (+) "The phosphorylation of $\langle e_1 \rangle$ by activated IRAK1 and $\langle e_2 \rangle$ could trigger and modulate the translocation of IRAKs from complex I to II";

3. (−) "The phosphorylation of Pellino2 by activated $\langle e_1 \rangle$ and $\langle e_2 \rangle$ could trigger and modulate the translocation of IRAKs from complex I to II";

The amount of human effort is still significant – the labels are provided by a domain expert through a process of manual annotation of sentences in the training corpus. However, manually labeling sentence examples is likely to be less demanding than manually devising a combination of extraction patterns. Also, many learning algorithms, when trained with this type of supervision, induce a set of extraction patterns whose weights are set such that the expected test error is minimized. In contrast, manually selecting an adequate set of weights can be a daunting task even for a domain expert. Examples of relation extractions approaches that learn from SIL supervision and that have motivated our research in this area are: the ELCS system developed by Rohit Kate and described in (Bunescu et al., 2005), the tree kernel method of Zelenko et al. (2003), and the dependency tree kernel approach of Culotta and Sorensen (2004).

In Sections 4.3 and 4.4 we describe two SIL approaches to relation extraction that differ with respect to the type of representation used for relation examples:

- In the first approach, features as simple as sequences of words around the two entity names can be used as extraction patterns. Additionally, the method can also utilize part-of-speech or phrase tags, if such information is available (Section 4.3).

- In the second approach, extraction patterns are created based on the shortest path between the two entity names in the dependency graph of the sentence. Compared to the first approach, this method requires a deeper syntactic analysis of the sentence examples (Section 4.4).

4.2.2 Multiple Instance Learning Supervision for RE

The amount of human effort could be further reduced if the only supervision required consisted of pairs of names of entities known to exhibit or not exhibit a particular relationship. Given a few pairs of well-known entities that clearly exhibit or do not exhibit a particular relation, such as `Acquired`(Google, YouTube) and `Not Acquired`(Yahoo, Microsoft), a search engine could be used on a very large corpus to find sentences that mention both of the entities in each of the pairs. The corpus is assumed to be large enough such that, although not all of the sentences for positive pairs will state the desired relationship, many of them will. Presumably, none of the sentences for negative pairs state the targeted relation. Each pair of entities will then be associated with a bag of extracted sentences, as illustrated below:

- (+) (Google, YouTube)
  - Search engine giant **Google** has bought video-sharing website **YouTube** in a controversial $1.6 billion deal.

  - The companies will merge **Google**'s search expertise with **YouTube**'s video expertise.
  - **Google** has acquired social media company, **YouTube** for $1.65 billion in a stock-for-stock transaction.

- (−) (Yahoo, Microsoft)
  - **Yahoo** is starting to look more like **Microsoft** and less like the innovative, unified service that got my loyalty in the first place.
  - Whatever it is, **Yahoo** is dashing in front, with **Microsoft** close behind.
  - **Yahoo** and **Microsoft** teamed up on October 12 to make their instant messaging software compatible.

Multiple instance learning is a machine learning framework that exploits this sort of weak supervision, in which a *positive bag* is a set of instances which is guaranteed to contain at least one positive example, and a *negative bag* is a set of instances all of which are negative. MIL was originally introduced to solve a problem in biochemistry (Dietterich et al., 1997); however, it has since been applied to problems in other areas such as classifying image regions in computer vision (Zhang et al., 2002), and text categorization (Andrews et al., 2003; Ray & Craven, 2005).

In Section 4.6 we extend our approach to relation extraction using support vector machines and string kernels (to be introduced in Section 4.3) to handle this weaker form of MIL supervision. This approach can sometimes be misled by textual features correlated with the specific entities in the few training pairs provided. Therefore, we also describe a method for weighting features in order to focus on those correlated with the target relation rather than with the individual entities.

4.3 SIL with a Subsequence Kernel for RE

The development of our first approach to relation extraction using single instance learning was motivated by previous work in text mining for protein-protein interactions. One of the early approaches to extracting interactions between proteins from biomedical abstracts is that of Blaschke and Valencia (2001, 2002). Their system is based on a set of manually developed rules, where each rule (or frame) is a sequence of words or part-of-speech (POS) tags and two protein-name tokens. Between every two adjacent words is a number indicating the maximum number of intervening words allowed when matching the rule to a sentence. An example rule is shown in Figure 4.1. A sentence matches the rule if and only if it satisfies the word constraints in the given order and respects the maximum length of the gaps between the words.

interaction of (3) $\langle e_1 \rangle$ (3) with (3) $\langle e_2 \rangle$

Figure 4.1: Sample extraction rule used in Blaschke's system.

Extraction using Longest Common Subsequences (ELCS) is a more recent method, developed by Rohit Kate and reported in (Bunescu et al., 2005), that automatically learns such rules. ELCS' rule representation is similar to that of Blaschke, except that it currently does not use POS tags, but allows disjunctions of words. An example rule learned by this system is shown in Figure 4.2. Words in square brackets separated by '|' indicate disjunctive lexical constraints, i.e. one of the given words must match the sentence at that position. The numbers in parentheses between adjacent constraints indicate the maximum number of unconstrained words allowed between the two.

- (7) interaction (0) [between | of] (5) $\langle e_1 \rangle$ (9) $\langle e_2 \rangle$ (17) .

Figure 4.2: Sample extraction rule used in Blaschke's system.

4.3.1 Capturing Relation Patterns with a String Kernel

Both Blaschke and ELCS do relation extraction based on a limited set of matching rules, where a rule is simply a sparse sequence of words or POS tags anchored on the two entity names. Therefore, the two methods share a common limitation: either through manual selection (Blaschke), or as a result of a greedy learning procedure (ELCS), they end up using only a subset of all possible anchored word sequences. Ideally, all such sequences of words would be used as features, with weights reflecting their relative accuracy. However explicitly creating for each sentence a vector with a position for each such feature is infeasible, due to the high dimensionality of the feature space. Nevertheless, we can exploit dual learning algorithms that process examples only via computing their dot-products, such as in Support Vector Machines (SVMs) (Vapnik, 1998; Cristianini & Shawe-Taylor, 2000). An SVM learner tries to find a hyperplane that separates positive from negative examples and at the same time maximizes the separation between them. This type of max-margin separator has been shown both theoretically and empirically to resist overfitting and to provide good generalization performance on unseen examples.

Computing the kernel (i.e. dot-product) between the features vectors associated with two relation examples amounts to calculating the number of common anchored subsequences between the two sentences. This is done efficiently by modifying the dynamic programming algorithm used in the string kernel of Lodhi et al. (2002) to account only for common sparse subsequences constrained to match at the two entity tokens. The feature space is further prunned down by utilizing the following property of natural language statements: when a sentence asserts a relationship between two entity mentions, it generally does this using one of the following four patterns:

• **[FB] F**ore–**B**etween: words before and between the two entity mentions are simultaneously used to express the relationship. Examples: 'interaction of $\langle e_1 \rangle$

with ⟨e_2⟩', 'activation of ⟨e_1⟩ by ⟨e_2⟩'.

• **[B] B**etween: only words between the two entities are essential for asserting the relationship. Examples: '⟨e_1⟩ interacts with ⟨e_2⟩', '⟨e_1⟩ is activated by ⟨e_2⟩'.

• **[BA] B**etween–**A**fter: words between and after the two entity mentions are simultaneously used to express the relationship. Examples: '⟨e_1⟩ – ⟨e_2⟩ complex', '⟨e_1⟩ and ⟨e_2⟩ interact'.

• **[M] M**odifier: the two entity mentions have no words between them. Examples: 'U.S. troops' (a ROLE:STAFF relation), 'Serbian general' (ROLE:CITIZEN).

While the first three patterns are sufficient to capture most cases of interactions between proteins, the last pattern is needed to account for various relationships expressed through noun-noun or adjective-noun compounds in other types of narrative.

Another observation is that all these patterns use at most 4 words to express the relationship (not counting the two entity names). Consequently, when computing the relation kernel, we restrict the counting of common anchored subsequences only to those having one of the four types described above, with a maximum word-length of 4. This type of feature selection leads not only to a faster kernel computation, but also to less overfitting, which results in increased accuracy. Notice that the rare cases in which the relationship is expressed using a pattern different from the four basic patterns above are still captured in this model by combinations of two or more patterns.

The patterns enumerated above are completely lexicalized and consequently their performance is limited by data sparsity. This can be alleviated by categorizing words into classes with varying degrees of generality, and then allowing patterns to use both words and their classes. Examples of word classes are POS tags and generalizations over POS tags such as Noun, Active Verb or Passive Verb. The entity type can also be used, if the word is part of a known named entity. Also,

if the sentence is segmented into syntactic chunks such as noun phrases (NP) or verb phrases (VP), the system may choose to consider only the head word from each chunk, together with the type of the chunk as another word class. Content words such as nouns and verbs can also be related to their synsets via WordNet. Patterns then can consist of sparse sequences of words, POS tags, generalized POS tags, entity and chunk types, or WordNet synsets. For example, 'Noun of $\langle e_1 \rangle$ by $\langle e_2 \rangle$' is an FB pattern based on words and general POS tags.

4.3.2 A Generalized Subsequence Kernel

Let $\Sigma_1, \Sigma_2, ..., \Sigma_k$ be a collection of disjoint feature spaces. Following the example in Section 4.3.1, Σ_1 could be the set of words, Σ_2 the set of POS tags, etc. Let $\Sigma_\times = \Sigma_1 \times \Sigma_2 \times ... \times \Sigma_k$ be the set of all possible feature vectors, where a feature vector would be associated with each position in a sentence. Given two feature vectors $x, y \in \Sigma_\times$, let $c(x, y)$ denote the number of common features between x and y. The next notation follows that introduced in (Lodhi et al., 2002). Thus, let s, t be two sequences over the finite set $\Sigma_\times$, and let $|s|$ denote the length of $s = s_1...s_{|s|}$. The sequence $s[i{:}j]$ is the contiguous subsequence $s_i...s_j$ of s. Let $\mathbf{i} = (i_1, ..., i_{|\mathbf{i}|})$ be a sequence of $|\mathbf{i}|$ indices in s, in ascending order. We define the length $l(\mathbf{i})$ *of the index sequence* $\mathbf{i}$ *in* s as $i_{|\mathbf{i}|} - i_1 + 1$. Similarly, $\mathbf{j}$ is a sequence of $|\mathbf{j}|$ indices in t.

Let $\Sigma_\cup = \Sigma_1 \cup \Sigma_2 \cup ... \cup \Sigma_k$ be the set of all possible features. We say that the sequence $u \in \Sigma_\cup^*$ is a (sparse) subsequence of s if there is a sequence of $|u|$ indices $\mathbf{i}$ such that $u_k \in s_{i_k}$, for all $k = 1, ..., |u|$. Equivalently, we write $u \prec s[\mathbf{i}]$ as a shorthand for the component-wise '$\in$' relationship between u and $s[\mathbf{i}]$.

Finally, let $K_n(s, t, \lambda)$ (Equation 4.1) be the number of weighted sparse subsequences u of length n common to s and t (i.e. $u \prec s[\mathbf{i}]$, $u \prec t[\mathbf{j}]$), where the weight

of u is $\lambda^{l(\mathbf{i})+l(\mathbf{j})}$, for some $\lambda \leq 1$.

$$K_n(s,t,\lambda) = \sum_{u \in \Sigma_{\cup}^n} \sum_{\mathbf{i}:u \prec s[\mathbf{i}]} \sum_{\mathbf{j}:u \prec t[\mathbf{j}]} \lambda^{l(\mathbf{i})+l(\mathbf{j})} \tag{4.1}$$

Let $\mathbf{i}$ and $\mathbf{j}$ be two index sequences of length n. By definition, for every k between 1 and n, $c(s_{i_k}, t_{j_k})$ returns the number of common features between s and t at positions i_k and j_k. If $c(s_{i_k}, t_{j_k}) = 0$ for some k, there are no common feature sequences of length n between $s[\mathbf{i}]$ and $t[\mathbf{j}]$. On the other hand, if $c(s_{i_k}, t_{j_k})$ is greater than 1, this means that there is more than one common feature that can be used at position k to obtain a common feature sequence of length n. Consequently, the number of common feature sequences of length n between $s[\mathbf{i}]$ and $t[\mathbf{j}]$, i.e. the size of the set $\{u \in \Sigma_{\cup}^n | u \prec s[\mathbf{i}], u \prec t[\mathbf{j}]\}$, is given by $\prod_{k=1}^{n} c(s_{i_k}, t_{j_k})$. Therefore, $K_n(s,t,\lambda)$ can be rewritten as in Equation 4.2:

$$K_n(s,t,\lambda) = \sum_{\mathbf{i}:|\mathbf{i}|=n} \sum_{\mathbf{j}:|\mathbf{j}|=n} \prod_{k=1}^{n} c(s_{i_k}, t_{j_k}) \lambda^{l(\mathbf{i})+l(\mathbf{j})} \tag{4.2}$$

We use λ as a penalty factor that downweights longer subsequences. For sparse subsequences, this means that wider gaps will be penalized more, which is exactly the desired behavior for the extraction patterns. Through them, we try to capture head-modifier dependencies that are important for relation extraction; for lack of reliable dependency information, the larger the word gap is between two words, the less confident we are in the existence of a head-modifier relationship between them.

To enable an efficient computation of K_n, we use the auxiliary function $K_n^{'}$ with a definition similar to K_n, the only difference being that it counts the length from the beginning of the particular subsequence u to the end of the strings s and t, as illustrated in Equation 4.3:

$$K_n^{'}(s,t,\lambda) = \sum_{u \in \Sigma_{\cup}^n} \sum_{\mathbf{i}:u \prec s[\mathbf{i}]} \sum_{\mathbf{j}:u \prec t[\mathbf{j}]} \lambda^{|s|+|t|-i_1-j_1+2} \tag{4.3}$$

An equivalent formula for $K_n^{'}(s,t,\lambda)$ is obtained by changing the exponent of λ from Equation 4.2 to $|s|+|t|-i_1-j_1+2$, as shown in Equation 4.4 below:

$$K_n^{'}(s,t,\lambda) = \sum_{\mathbf{i}:|\mathbf{i}|=n} \sum_{\mathbf{j}:|\mathbf{j}|=n} \prod_{k=1}^{n} c(s_{i_k}, t_{j_k}) \lambda^{|s|+|t|-i_1-j_1+2} \tag{4.4}$$

Based on all definitions above, K_n is computed in $O(kn|s||t|)$ time, by modifying the recursive computation of Lodhi et al. (2002) with the new factor $c(x,y)$, as shown in Figure 4.3. As in (Lodhi et al., 2002), the complexity of computing $K_i^{'}(s,t)$ is reduced to $O(|s||t|)$ by first evaluating another auxiliary factor $K_i^{''}(s,t)$. In Figure 4.3, the sequence sx is the result of appending x to s (with ty defined in a similar way). To avoid clutter, the parameter λ is not shown in the argument list of K and K', unless it is instantiated to a specific constant.

$$\begin{aligned}
K_0^{'}(s,t) &= 1, \ for\ all\ s,t \\
K_i^{'}(s,t) &= 0, \ if\ \min(|s|,|t|) < i \\
K_i^{''}(s,\emptyset) &= 0, \ for\ all\ i,s \\
K_i^{''}(sx,ty) &= \lambda K_i^{''}(sx,t) + \lambda^2 K_{i-1}^{'}(s,t) \cdot c(x,y) \\
K_i^{'}(sx,t) &= \lambda K_i^{'}(s,t) + K_i^{''}(sx,t) \\
K_n(s,t) &= 0, \ if\ \min(|s|,|t|) < n \\
K_n(sx,t) &= K_n(s,t) + \sum_j \lambda^2 K_{n-1}^{'}(s, t[1:j-1]) \cdot c(x, t[j])
\end{aligned}$$

Figure 4.3: Computation of subsequence kernel.

4.3.3 Computing the Subsequence Relation Kernel

As described at the beginning of Section 4.3, the input consists of a set of sentences, where each sentence contains exactly two entities (protein names in the case of interaction extraction). In Figure 4.4 we show the segments that will be used for computing the relation kernel between two example sentences s and t. In sentence s for instance, x_1 and x_2 are the two entities, s_f is the sentence segment before x_1, s_b is the segment between x_1 and x_2, and s_a is the sentence segment after x_2. For convenience, we also include the auxiliary segment $s_b^{'} = x_1 s_b x_2$, whose span is computed as $l(s_b^{'}) = l(s_b) + 2$ (in all length computations, we consider x_1 and x_2 as contributing one unit only).

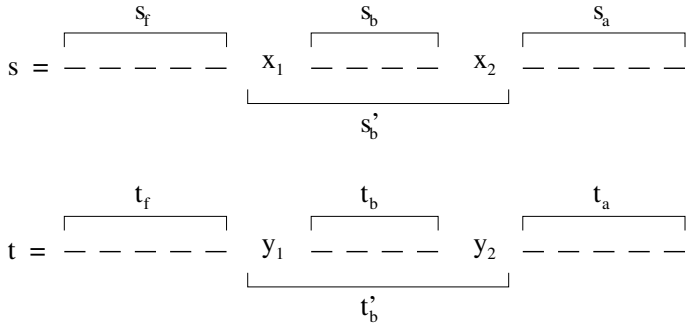


Figure 4.4: Sentence segments.

The relation kernel computes the number of common patterns between two sentences s and t, where the set of patterns is restricted to the four types introduced in Section 4.3.1. Therefore, the kernel $rK(s,t)$ is expressed as the sum of four sub-kernels: $fbK(s,t)$ counting the number of common fore–between patterns, $bK(s,t)$ for between patterns, $baK(s,t)$ for between–after patterns, and $mK(s,t)$ for modifier patterns, as in Figure 4.5. The symbol $\mathbb{1}$ is used there as a shorthand for the indicator function, which is 1 if the argument is true, and 0 otherwise.

The first three sub-kernels include in their computation the counting of com-

$$
\begin{aligned}
rK(s,t) &= fbK(s,t) + bK(s,t) + baK(s,t) + mK(s,t) \\
bK_i(s,t) &= K_i(s_b, t_b, 1) \cdot c(x_1, y_1) \cdot c(x_2, y_2) \cdot \lambda^{l(s_b^{'}) + l(t_b^{'})} \\
fbK(s,t) &= \sum_{i,j} bK_i(s,t) \cdot K_j^{'}(s_f, t_f), \quad 1 \leq \mathrm{i},\ 1 \leq \mathrm{j},\ \mathrm{i} + \mathrm{j} < \mathrm{fb}_{\max} \\
bK(s,t) &= \sum_{i} bK_i(s,t), \quad 1 \leq \mathrm{i} \leq \mathrm{b}_{\max} \\
baK(s,t) &= \sum_{i,j} bK_i(s,t) \cdot K_j^{'}(s_a^{-}, t_a^{-}), \quad 1 \leq \mathrm{i},\ 1 \leq \mathrm{j},\ \mathrm{i} + \mathrm{j} < \mathrm{ba}_{\max} \\
mK(s,t) &= \mathbb{1}(s_b = \emptyset) \cdot \mathbb{1}(t_b = \emptyset) \cdot c(x_1, y_1) \cdot c(x_2, y_2) \cdot \lambda^{2+2},
\end{aligned}
$$

Figure 4.5: Computation of subsequence relation kernel.

mon subsequences between $s_b^{'}$ and $t_b^{'}$. In order to speed up the computation, all these common counts are calculated separately in bK_i, which is defined as the number of common subsequences of length i between $s_b^{'}$ and $t_b^{'}$, anchored at x_1/x_2 and y_1/y_2 respectively (i.e. constrained to start at x_1 in $s_b^{'}$ and y_1 in $t_b^{'}$, and to end at x_2 in $s_b^{'}$ and y_2 in $t_b^{'}$). Then fbK simply counts the number of subsequences that match j positions before the first entity and i positions between the entities, constrained to have length less than a constant fb_{max}. To obtain a similar formula for baK we simply use the reversed (mirror) version of segments s_a and t_a (e.g. s_a^{-} and t_a^{-}). In Section 4.3.1 we constrained the three subsequence patterns to use at most 4 words to express a relation, therefore the constants fb_{max}, b_{max} and ba_{max} are set to 4. Kernels K and $K^{'}$ are computed using the procedure described in Section 4.3.2.

4.4 SIL with a Dependency Path Kernel for RE

The pattern examples from Section 4.3.1 show the two entity mentions, together with the set of words that are relevant for their relationship. A closer analysis of these examples reveals that all relevant words form a shortest path between the

two entities in a graph structure where edges correspond to relations between a head word and its dependents. Figure 4.6 shows the full dependency graphs for two sentences from the Automated Content Extraction (ACE) newspaper corpus (NIST, 2000), in which words are represented as nodes and word-word dependencies are represented as directed edges. A subset of these word-word dependencies capture the predicate-argument relations present in the sentence. Arguments are connected to their target predicates either directly through an arc pointing to the predicate ('troops → raided'), or indirectly through a preposition or infinitive particle ('warning ← to ← stop'). Other types of word-word dependencies account for modifier-head relationships present in adjective-noun compounds ('several → stations'), noun-noun compounds ('pumping → stations'), or adverb-verb constructions ('recently → raided').

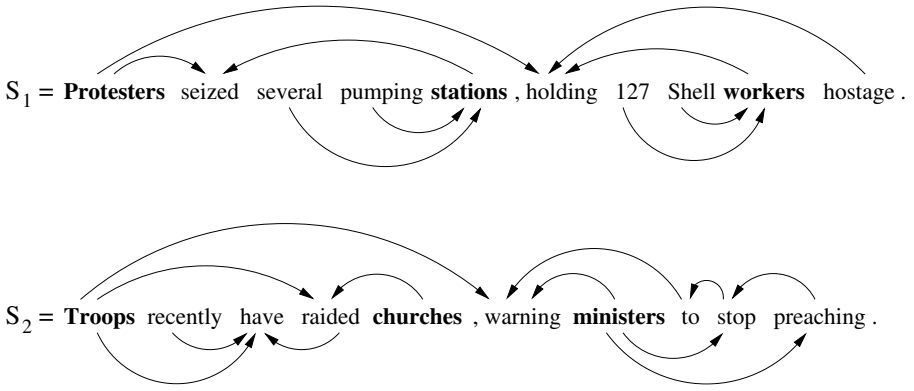


Figure 4.6: Sentences as dependency graphs.

Dependency representations of language structure have a long tradition (Hudson, 1984). Word-word dependencies are typically categorized in two classes as follows:

- **[Local Dependencies]** These correspond to local predicate-argument (or head-modifier) constructions such as 'troops → raided', or 'pumping → sta-

tions' in Figure 4.6.

- **[Non-local Dependencies]** Long-distance dependencies arise due to various linguistic constructions such as coordination, extraction, raising and control. In Figure 4.6, among non-local dependencies are 'troops → warning', or 'ministers → preaching'.

A Context Free Grammar (CFG) parser can be used to extract local dependencies (Collins, 1999), which for each sentence form a dependency tree. Mildly context sensitive formalisms such as Combinatory Categorial Grammar (CCG) (Steedman, 2000) model word-word dependencies more directly and can be used to extract both local and long-distance dependencies, resulting in a directed acyclic graph, as illustrated in Figure 4.6.

4.4.1 The Shortest Path Hypothesis

If e_1 and e_2 are two entities mentioned in the same sentence such that they are observed to be in a relationship R, then most of the contribution of the sentence dependency graph to establishing the relationship $R(e_1, e_2)$ is concentrated in the shortest path between e_1 and e_2 in the undirected version of the dependency graph.

If entities e_1 and e_2 are arguments of the same predicate, then the shortest path between them will pass through the predicate, which may be connected directly to the two entities, or indirectly through prepositions. If e_1 and e_2 belong to different predicate-argument structures that share a common argument, then the shortest path will pass through this argument. This is the case with the shortest path between 'stations' and 'workers' in Figure 4.6, passing through 'protesters', which is an argument common to both predicates 'holding' and 'seized'. In Table 4.1, we show the paths corresponding to the four relation instances encoded in the ACE corpus for the two sentences from Figure 4.6. All these paths support the LOCATED relationship. For the first path, it is reasonable to infer that if a PERSON entity

(e.g. 'protesters') is doing some action (e.g. 'seized') to a FACILITY entity (e.g. 'station'), then the PERSON entity is LOCATED at that FACILITY entity. The second path captures the fact that the same PERSON entity (e.g. 'protesters') is doing two actions (e.g. 'holding' and 'seized') , one action to a PERSON entity (e.g. 'workers'), and the other action to a FACILITY entity (e.g. 'station'). A reasonable inference in this case is that the 'workers' are LOCATED at the 'station'.

Relation Instance	Shortest Undirected Path in Dependency Graph
S_1:protesters AT stations	**protesters** → seized ← **stations**
S_1:workers AT stations	**workers** → holding ← protesters → seized ← **stations**
S_2:troops AT churches	**troops** → raided ← **churches**
S_2:ministers AT churches	**ministers** → warning ← troops → raided ← **churches**

Table 4.1: Shortest Path representation of relations.

In Figure 4.7, we show three more examples of the LOCATED (AT) relationship as dependency paths created from one or two predicate-argument structures. The second example is an interesting case, as it illustrates how annotation decisions are accommodated in our approach. Using a reasoning similar with that from the previous paragraph, it is reasonable to infer that 'troops' are LOCATED in 'vans', and that 'vans' are LOCATED in 'city'. However, because 'vans' is not an ACE markable, it cannot participate in an annotated relationship. Therefore, 'troops' is annotated as being LOCATED in 'city', which makes sense due to the transitivity of the relation LOCATED. In our approach, this leads to shortest paths that pass through two or more predicate-argument structures.

The last relation example is a case where there exist multiple shortest paths in the dependency graph between the same two entities – there are actually two different paths, with each path replicated into three similar paths due to coordination. Our current approach considers only one of the shortest paths, nevertheless it could be extended to use all of them as multiple sources of evidence for relation extraction.

(1) He had no regrets for **his** actions in **Brcko**.

his → actions ← in ← **Brcko**

(2) U.S. **troops** today acted for the first time to capture an alleged Bosnian war criminal, rushing from unmarked vans parked in the northern Serb-dominated **city** of Bijeljina.

troops → rushing ← from ← vans → parked ← in ← **city**

(3) Jelisic created an atmosphere of terror at the **camp** by killing, abusing and threatening the **detainees**.

detainees → killing ← Jelisic → created ← at ← **camp**
detainees → abusing ← Jelisic → created ← at ← **camp**
detainees → threatening ← Jelisic → created ← at ← **camp**
detainees → killing → by → created ← at ← **camp**
detainees → abusing → by → created ← at ← **camp**
detainees → threatening → by → created ← at ← **camp**

Figure 4.7: Relation examples.

There may be cases where e_1 and e_2 belong to predicate-argument structures that have no argument in common. However, because the dependency graph is always connected, we are guaranteed to find a shortest path between the two entities. In general, we shall find a shortest sequence of predicate-argument structures with target predicates $P_1, P_2, ..., P_n$ such that e_1 is an argument of P_1, e_2 is an argument of P_n, and any two consecutive predicates P_i and P_{i+1} share a common argument (where by "argument" we mean both arguments and complements).

4.4.2 Learning with Dependency Paths

The shortest path between two entities in a dependency graph offers a very condensed representation of the information needed to assess their relationship. A dependency path is represented as a sequence of words interspersed with arrows that indicate the orientation of each dependency, as illustrated in Table 4.1. These

paths however are completely lexicalized and consequently their performance will be limited by data sparsity. The solution is to allow paths to use both words and their word classes, similar with the approach taken for the subsequence patterns in Section 4.3.1.

The set of features can then be defined as a Cartesian product over words and word classes, as illustrated in Figure 4.8 for the dependency path between 'protesters' and 'station' in sentence S_1. In this representation, sparse or contiguous subsequences of nodes along the lexicalized dependency path (i.e. path fragments) are included as features simply by replacing the rest of the nodes with their corresponding generalizations.

$$\begin{bmatrix} \text{protesters} \\ \text{NNS} \\ \text{Noun} \\ \textsc{Person} \end{bmatrix} \times [\rightarrow] \times \begin{bmatrix} \text{seized} \\ \text{VBD} \\ \text{Verb} \end{bmatrix} \times [\leftarrow] \times \begin{bmatrix} \text{stations} \\ \text{NNS} \\ \text{Noun} \\ \textsc{Facility} \end{bmatrix}$$

Figure 4.8: Feature generation from dependency path.

Examples of features generated by Figure 4.8 are "protesters → seized ← stations", "Noun → Verb ← Noun", "PERSON → seized ← FACILITY", or "PERSON → Verb ← FACILITY". The total number of features generated by this dependency path is $4 \times 1 \times 3 \times 1 \times 4$.

For verbs and nouns (and their respective word classes) occurring along a dependency path we also use an additional suffix '(-)' to indicate a negative polarity item. In the case of verbs, this suffix is used when the verb (or an attached auxiliary) is modified by a negative polarity adverb such as 'not' or 'never'. Nouns get the negative suffix whenever they are modified by negative determiners such as 'no', 'neither' or 'nor'. For example, the phrase "He never went to Paris" is associated with the dependency path "He → went(-) ← to ← Paris". A more principled approach that we investigate in future work is to augment the dependency paths

which dependency trees rooted at content words along path, and treat each rooted tree as a generalized word class. One advantage of this extension is that negative polarity items would be automatically learned from the training examples.

As in Section 4.3, we use kernel SVMs in order to avoid working explicitly with high-dimensional dependency path feature vectors. Computing the dot-product (i.e. kernel) between two relation examples amounts to calculating the number of common features (i.e. paths) between the two examples. If $\boldsymbol{x} = x_1 x_2 ... x_m$ and $\boldsymbol{y} = y_1 y_2 ... y_n$ are two relation examples, where x_i denotes the set of word classes corresponding to position i (as in Figure 4.8), then the number of common features between $\boldsymbol{x}$ and $\boldsymbol{y}$ is computed as in Equation 4.5.

$$K(\mathbf{x}, \mathbf{y}) = \mathbb{1}(m = n) \cdot \prod_{i=1}^{n} c(x_i, y_i) \tag{4.5}$$

where $c(x_i, y_i) = |x_i \cap y_i|$ is the number of common word classes between x_i and y_i.

This is a simple kernel, whose computation takes $O(n)$ time. If the two paths have different lengths, they correspond to different ways of expressing a relationship – for instance, they may pass through a different number of predicate argument structures. Consequently, the kernel is defined to be 0 in this case. Otherwise, it is the product of the number of common word classes at each position in the two paths. As an example, let us consider two instances of the LOCATED relationship, and their corresponding dependency paths:

1. '**his** actions in **Brcko**' (**his** → actions ← in ← **Brcko**).

2. '**his** arrival in **Beijing**' (**his** → arrival ← in ← **Beijing**).

Their representation as a sequence of sets of word classes is given by:

1. $\boldsymbol{x} = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7]$, where x_1 = {his, PRP, PERSON}, x_2 = {→}, x_3 = {actions, NNS, Noun}, x_4 = {←}, x_5 = {in, IN}, x_6 = {←}, x_7 = {Brcko, NNP, Noun, LOCATION}

2. $\boldsymbol{y} = [y_1\ y_2\ y_3\ y_4\ y_5\ y_6\ y_7]$, where y_1 = {his, PRP, PERSON}, y_2 = {$\rightarrow$}, y_3 = {arrival, NN, Noun}, y_4 = {$\leftarrow$}, y_5 = {in, IN}, y_6 = {$\leftarrow$}, y_7 = {Beijing, NNP, Noun, LOCATION}

Based on the formula from Equation 4.5, the kernel is computed as $K(\boldsymbol{x}, \boldsymbol{y}) = 3 \times 1 \times 1 \times 1 \times 2 \times 1 \times 3 = 18$.

4.5 SIL Experimental Results

The two relation kernels described above are evaluated on the task of extracting relations from two corpora with different types of narrative, which are described in more detail in the following sections. In both cases, we assume that the entities and their labels are known. All preprocessing steps – sentence segmentation, tokenization, POS tagging and chunking – were performed using the OpenNLP[1] package. As explained in Section 4.2.1, if a sentence contains n entities ($n \geq 2$), it is replicated into $\binom{n}{2}$ sentence examples, each containing only two entities.

The dependency graph that is input to the shortest path dependency kernel is obtained from two different parsers:

- The CCG parser introduced of Hockenmaier and Steedman (2002)[2] outputs a list of functor-argument dependencies, from which head-modifier dependencies are obtained using a straightforward, deterministic procedure.
- Head-modifier dependencies can be easily extracted from the full parse output of Collins' CFG parser (Collins, 1997), in which every non-terminal node is annotated with head information.

The relation kernels are used in conjunction with SVM learning in order to find a decision hyperplane that best separates the positive examples from negative

[1]URL: http://opennlp.sourceforge.net
[2]URL:http://www.ircs.upenn.edu/~juliahr/Parser/

examples. We modified the LibSVM[3] package by plugging in the kernels described above. The factor λ in the subsequence kernel is set to 0.75. The performance is measured using *precision* (percentage of correctly extracted relations out of the total number of relations extracted), *recall* (percentage of correctly extracted relations out of the total number of relations annotated in the corpus), and *F-measure* (the harmonic mean of *precision* and *recall*).

4.5.1 Interaction Extraction from AIMed

We did comparative experiments on the AIMed corpus, which has been previously used for training the protein interaction extraction systems in (Bunescu et al., 2005). 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 4,084 protein references and around 1,000 tagged interactions in this dataset. We assume that the abstracts have already been tagged for protein names – for evaluation we used the gold standard manual annotations.

The following systems are evaluated on the task of extracting protein interactions from AIMed:

• **[Manual]**: We report the performance of the rule-based system of Blaschke and Valencia (2001, 2002).

• **[ELCS]**: We report the 10-fold cross-validated results from (Bunescu et al., 2005) as a Precision-Recall (PR) graph.

• **[SSK]**: The subsequence kernel is trained and tested on the same splits as ELCS. In order to have a fair comparison with the other two systems, SSK was constrained to use only lexical information.

• **[SPK]**: This is the shortest path dependency kernel, using the head-modifier dependencies extracted by Collins' syntactic parser. The kernel is trained

[3]URL:http://www.csie.ntu.edu.tw/~cjlin/libsvm/

and tested on the same 10 splits as ELCS and SSK.

The Precision-Recall curves that show the trade-off between these metrics are obtained by varying a threshold on the minimum acceptable extraction confidence, based on the probability estimates from LibSVM. The results, summarized in Figure 4.9(a), show that the subsequence kernel outperforms the other three systems, with a substantial gain. The syntactic parser, which is originally trained on a newspaper corpus, builds less accurate dependency structures for the biomedical text. This is reflected in a significantly reduced accuracy for the dependency kernel. Recent efforts aimed at the annotation of biomedical corpora with syntactic structures (Tateisi et al., 2005) may lead to the development of syntactic parsers that output more accurate dependency structures, which in turn may yield an increased performance for the dependency kernel approach to relation extraction.

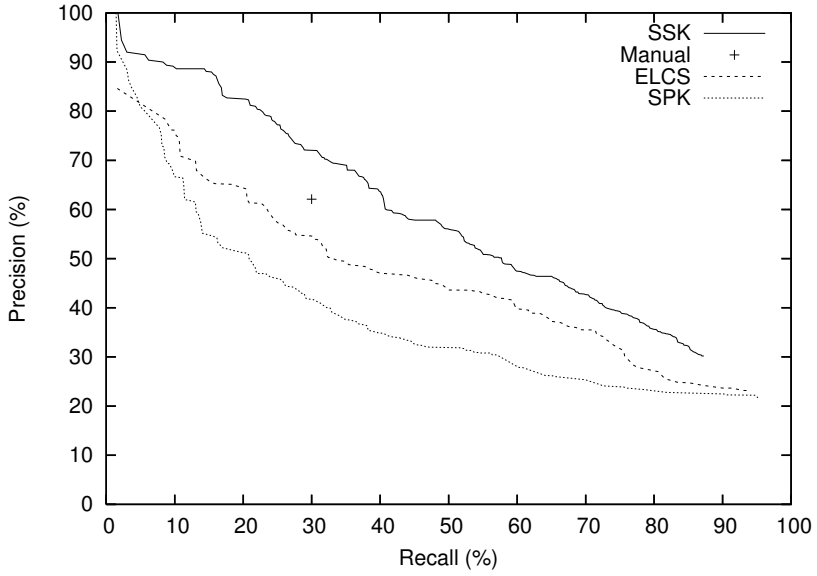


Figure 4.9: Precision-Recall curves for protein interaction extractors.

4.5.2 Relation Extraction from ACE

The two kernels are also evaluated on the task of extracting top-level relations from the ACE corpus (NIST, 2000), the version used for the September 2002 evaluation. The training part of this dataset consists of 422 documents, with a separate set of 97 documents reserved for testing. This version of the ACE corpus contains three types of annotations: coreference, named entities and relations. There are five types of entities – PERSON, ORGANIZATION, FACILITY, LOCATION, and GEO-POLITICAL ENTITY – which can participate in five general, top-level relations: ROLE, PART, LOCATED, NEAR, and SOCIAL. In total, there are 7,646 intra-sentential relations, of which 6,156 are in the training data and 1,490 in the test data.

A recent approach to extracting relations is described by Culotta and Sorensen (2004). The authors use a generalized version of the tree kernel of Zelenko et al. (2003) to compute a kernel over relation examples, where a relation example consists of the smallest dependency tree containing the two entities of the relation. Precision and recall values are reported for the task of extracting the 5 top-level relations in the ACE corpus under two different scenarios:

– **[S1]** This is the classic setting: one multi-class SVM is learned to discriminate among the 5 top-level classes, plus one more class for the no-relation cases.

– **[S2]** One binary SVM is trained for *relation detection*, meaning that all positive relation instances are combined into one class. The thresholded output of this binary classifier is used as training data for a second multi-class SVM, trained for *relation classification*.

The subsequence kernel (SSK) is trained under the first scenario, to recognize the same 5 top-level relation types. While for protein interaction extraction only the lexicalized version of the kernel was used, here we utilize more features, corresponding to the following feature spaces: Σ_1 is the word vocabulary, Σ_2 is the set of POS tags, Σ_3 is the set of generic POS tags, and Σ_4 contains the 5 entity types.

Chunking information is used as follows: all (sparse) subsequences are created exclusively from the chunk heads, where a head is defined as the last word in a chunk. The same criterion is used for computing the length of a subsequence – all words other than head words are ignored. This is based on the observation that in general words other than the chunk head do not contribute to establishing a relationship between two entities outside of that chunk. One exception is when both entities in the example sentence are contained in the same chunk. This happens very often due to noun-noun ('U.S. troops') or adjective-noun ('Serbian general') compounds. In these cases, the chunk is allowed to contribute both entity heads.

The shortest-path dependency kernel (SPK) is trained under both scenarios. The dependencies are extracted using either Hockenmaier's CCG parser (SPK-CCG) (Hockenmaier & Steedman, 2002), or Collins' CFG parser (SPK-CFG) (Collins, 1997). To avoid numerical problems, the dependency paths are constrained to pass through at most 10 words (as observed in the training data) by setting the kernel to 0 for longer paths. The alternative solution of normalizing the kernel leads to a slight decrease in accuracy. The fact that longer paths have larger kernel scores in the unnormalized version does not pose a problem because, by definition, paths of different lengths correspond to disjoint sets of features. Consequently, the SVM algorithm will induce lower weights for features occurring in longer paths, resulting in a linear separator that works irrespective of the size of the dependency paths.

Table 4.2 summarizes the performance of the two relation kernels on the ACE corpus. For comparison, we also show the results presented in (Culotta & Sorensen, 2004) for their best performing kernel K4 (a sum between a bag-of-words kernel and a tree dependency kernel) under both scenarios.

The subsequence kernel performs better than the dependency tree kernel K4 – even though SSK does not use dependency information, its features preserve the word order observed in the training examples, unlike the simple bag-of-words kernel

(Scenario) Method	Precision	Recall	F-measure
(S1) K4	70.3	26.3	38.0
(S1) SSK	73.9	35.2	47.7
(S1) SPK-CCG	67.5	37.2	48.0
(S1) SPK-CFG	**71.1**	**39.2**	**50.5**
(S2) K4	67.1	35.0	45.8
(S2) SPK-CCG	63.7	41.4	50.2
(S2) SPK-CFG	**65.5**	**43.8**	**52.5**

Table 4.2: Extraction Performance on ACE.

baseline from (Culotta & Sorensen, 2004). This result validates our prior belief that word order is an important factor in relation extraction.

The shortest-path dependency kernels outperform the dependency kernel of Culotta and Sorensen (2004) in both scenarios, with a more substantial gain for SP-CFG. This confirms our intuition that increased extraction performance can be obtained by focusing the features space only on those dependency structures that are relevant to relation extraction. An error analysis revealed that Collins' parser was better at capturing local dependencies, hence the increased accuracy of SP-CFG over SP-CCG. Another advantage of shortest-path dependency kernels is that their training and testing are very fast – this is due to representing the sentence as a chain of dependencies on which a fast kernel can be computed. All the four SP kernels from Table 4.2 take between 2 and 3 hours to train and test on a 2.6GHz Pentium IV machine.

As expected, the newspaper articles from ACE are less prone to parsing errors than the biomedical articles from AIMed. Consequently, the extracted dependency structures are more accurate, leading to an improved accuracy for the dependency kernel when compared to the subsequence kernel – in contrast to the results on the biomedical domain.

4.6 An MIL Approach to Relation Extraction

The two SIL methods described so far – the subsequence kernel in Section 4.3 and the dependency path kernel in Section 4.4 – require the annotation of large corpora with examples of the relations to be extracted, which is an expensive and tedious process. In this section, we introduce a supervised learning approach to relation extraction that requires only a handful of training examples in conjunction with a very large corpus.

4.6.1 Problem Definition

We address the task of learning a relation extraction system targeted to a fixed binary relationship R. The only supervision given to the learning algorithm is a small set of pairs of named entities that are known to belong (positive) or not belong (negative) to the given relationship. Table 4.3 shows four positive and two negative example pairs for the corporate acquisition relationship. For each pair, a bag of sentences containing the two arguments can be extracted from a corpus of text documents. The corpus is assumed to be sufficiently large and diverse such that, if the pair is positive, it is highly likely that the corresponding bag contains at least one sentence that explicitly asserts the relationship R between the two arguments. In Section 4.7.2 we describe a method for extracting bags of relevant sentences from the web.

+/−	**Arg** a_1	**Arg** a_2
+	Google	YouTube
+	Adobe Systems	Macromedia
+	Viacom	DreamWorks
+	Novartis	Eon Labs
−	Yahoo	Microsoft
−	Pfizer	Teva

Table 4.3: Corporate Acquisition Pairs.

Using a limited set of entity pairs (e.g. those in Table 4.3) and their associated bags as training data, the aim is to induce a relation extraction system that can reliably decide whether two entities mentioned in the same sentence exhibit the target relationship or not. In particular, when tested on the example sentences from Figure 4.10, the system should classify S_1, S_3,and S_4 as positive, and S_2 and S_5 as negative.

$+/S_1$: Search engine giant **Google** has bought video-sharing website **YouTube** in a controversial $1.6 billion deal.

$-/S_2$: The companies will merge **Google**'s search expertise with **YouTube**'s video expertise, pushing what executives believe is a hot emerging market of video offered over the Internet.

$+/S_3$: **Google** has acquired social media company, **YouTube** for $1.65 billion in a stock-for-stock transaction as announced by Google Inc. on October 9, 2006.

$+/S_4$: Drug giant **Pfizer Inc.** has reached an agreement to buy the private biotechnology firm **Rinat Neuroscience Corp.**, the companies announced Thursday.

$-/S_5$: He has also received consulting fees from Alpharma, Eli Lilly and Company, **Pfizer**, Wyeth Pharmaceuticals, **Rinat Neuroscience**, Elan Pharmaceuticals, and Forest Laboratories.

Figure 4.10: Sentence examples.

As formulated above, the learning task can be seen as an instance of *multiple instance learning*. However, there are important properties that set it apart from problems previously considered in MIL. The most distinguishing characteristic is that the number of bags is very small, while the average size of the bags is very large.

4.6.2 The MIL Formulation

Since its introduction by Dietterich et al. (1997), an extensive and quite diverse set of methods have been proposed for solving the MIL problem. For the task of relation extraction, we consider only MIL methods where the decision function can be expressed in terms of kernels computed between bag instances. This choice was motivated by the comparatively high accuracy obtained by kernel-based SVMs when applied to various natural language tasks, and in particular to relation extraction.

Gartner et al. (2002) adapted SVMs to the MIL setting using various multi-instance kernels. Two of these – the normalized set kernel, and the statistic kernel – have been experimentally compared to other methods by Ray and Craven (2005), with competitive results. Alternatively, a simple approach to MIL is to transform it into a standard supervised learning problem by labeling all instances from positive bags as positive. An interesting outcome of the study conducted by Ray and Craven (2005) was that, despite the class noise in the resulting positive examples, such a simple approach often obtains competitive results when compared against other more sophisticated MIL methods. Based on this observation, we decided to transform the MIL problem into a standard supervised SVM problem as illustrated in Figure 4.11. In this formulation, $\mathcal{X}$ is the set of bags used for training, $\mathcal{X}_p \subseteq \mathcal{X}$ the set of positive bags, and $\mathcal{X}_n \subseteq \mathcal{X}$ the set of negative bags. For any instance $x \in X$ from a bag $X \in \mathcal{X}$, $\phi(x)$ is the implicit feature vector representation of x.

minimize:

$$\mathbf{J}(w, b, \xi) = \tfrac{1}{2}\,||w||^2 + \frac{C}{L}\left(c_p \frac{L_n}{L} \sum_{X \in \mathcal{X}_p} \sum_{x \in X} \xi_x + c_n \frac{L_p}{L} \sum_{X \in \mathcal{X}_n} \sum_{x \in X} \xi_x\right)$$

subject to:

$$w\,\phi(x) + b \geq +1 - \xi_x, \;\; \forall x \in X \in \mathcal{X}_p$$
$$w\,\phi(x) + b \leq -1 + \xi_x, \;\; \forall x \in X \in \mathcal{X}_n$$
$$\xi_x \geq 0$$

Figure 4.11: SVM Optimization Problem.

The capacity control parameter C is normalized by the total number of instances $L = L_p + L_n = \sum_{X \in \mathcal{X}_p} |X| + \sum_{X \in \mathcal{X}_n} |X|$, so that it remains independent of the size of the dataset. The additional non-negative parameter c_p ($c_n = 1 - c_p$) controls the relative influence that false negative vs. false positive errors have on the value of the objective function. Because not all instances from positive bags are real positive instances, it makes sense to have false negative errors be penalized less than false positive errors (i.e. $c_p < 0.5$).

In the dual formulation of the optimization problem from Figure 4.11, bag instances appear only inside dot products of the form $K(x_1, x_2) = \phi(x_1)\phi(x_2)$, which makes it possible to use kernels.

The training bags consist of sentences extracted from online documents, using the methodology described in Section 4.7.2. Parsing web documents in order to obtain a syntactic analysis often gives unreliable results – the type of narrative can vary greatly from one web document to another, and sentences with grammatical errors are frequent. Therefore, we decided to instantiate K to a modified version of the subsequence kernel from Section 4.3, which does not require syntactic information. As described in Section 4.3, the kernel value corresponds to the number of common subsequences of tokens between two sentences. The subsequences are constrained to be "anchored" at the two entity names, and there is a maximum number of tokens that can appear in a sequence. For example, a subsequence feature for the sentence S_1 in Figure 4.10 is $\tilde{s}$ = *"*$\langle e_1 \rangle$ ... *bought* ... $\langle e_2 \rangle$ ... *in* ... *billion* ... *deal"*, where $\langle e_1 \rangle$ and $\langle e_2 \rangle$ are generic placeholders for the two entity names. Let $s = w_1 w_2 \ldots w_k$ be a sequence of k words, and $\tilde{s} = w_1\, g_1\, w_2\, g_2 \ldots w_{k-1}\, g_{k-1}\, w_k$ a matching subsequence in a relation example, where g_i stands for any sequence of words between w_i and w_{i+1}. Then the sequence s will be represented in the relation example as a feature with weight computed as $\tau(s) = \lambda^{g(\tilde{s})}$. The parameter λ controls the magnitude of the gap penalty, where $g(\tilde{s}) = \sum_i |g_i|$ is the total gap.

Many relations, like the ones that we explore in the experimental evaluation, cannot be expressed without using at least one content word. We therefore modified the kernel computation to optionally ignore subsequence patterns formed exclusively of stop words and punctuation signs. In Section 4.6.4, we introduce a new weighting scheme, wherein a weight is assigned to every token. Correspondingly, every sequence feature will have an additional multiplicative weight, computed as the product of the weights of all the tokens in the sequence. The aim of this new weighting scheme, as detailed in the next section, is to eliminate the bias caused by the special structure of the relation extraction MIL problem.

4.6.3 Two Types of Bias

As already hinted at the end of Section 4.6.1, there is one important property that distinguishes the current MIL setting for relation extraction from other MIL problems: the training dataset contains very few bags, and each bag can be very large. Consequently, an application of the learning model described in Section 4.6.2 is bound to be affected by the following two types of bias:

■ [**Type I Bias**] By definition, all sentences inside a bag are constrained to contain the same two arguments. Words that are semantically correlated with either of the two arguments are likely occur in many sentences. For example, consider the sentences S_1 and S_2 from the bag associated with "Google" and "YouTube" (as shown in Figure 4.10). They both contain the words "search" – highly correlated with "Google", and "video" – highly correlated with "YouTube", and it is likely that a significant percentage of sentences in this bag contain one of the two words (or both). The two entities can be mentioned in the same sentence for reasons other than the target relation R, and these noisy training sentences are likely to contain words that correlated with the two entities, without any relationship to R. A learning model where the features are based on words, or word sequences, is going

to give too much weight to words or combinations of words that are correlated with either of individual arguments. This overweighting will adversely affect extraction performance through an increased number of errors. A method for eliminating this type of bias is introduced in Section 4.6.4.

■ **[Type II Bias]** While Type I bias is due to words that are correlated with the arguments of a relation instance, the Type II bias is caused by words that are specific to the relation instance itself. Using FrameNet terminology (Baker et al., 1998), these correspond to instantiated frame elements. For example, the corporate acquisition frame can be seen as a subtype of the "Getting" frame in FrameNet. The *core* elements in this frame are the *Recipient* (e.g. Google) and the *Theme* (e.g. YouTube), which for the acquisition relationship coincide with the two arguments. They do not contribute any bias, since they are replaced with the generic tags $\langle e_1 \rangle$ and $\langle e_2 \rangle$ in all sentences from the bag. There are however other frame elements – *peripheral*, or *extra-thematic* – that can be instantiated with the same value in many sentences. In Figure 4.10, for instance, sentence S_3 contains two non-core frame elements: the *Means* element (e.g "in a stock-for-stock transaction") and the *Time* element (e.g. "on October 9, 2006"). Words from these elements, like "stock", or "October", are likely to occur very often in the Google-YouTube bag, and because the training dataset contains only a few other bags, subsequence patterns containing these words will be given too much weight in the learned model. This is problematic, since these words can appear in many other frames, and thus the learned model is likely to make errors. Instead, we would like the model to focus on words that trigger the target relationship (in FrameNet, these are the lexical units associated with the target frame).

4.6.4 A Solution for Type I Bias

In order to account for how strongly the words in a sequence are correlated with either of the individual arguments of the relation, we modify the formula for the sequence weight $\tau(s)$ by factoring in a weight $\tau(w)$ for each word in the sequence, as illustrated in Equation 4.6.

$$\tau(s) = \lambda^{g(\bar{s})} \cdot \prod_{w \in s} \tau(w) \tag{4.6}$$

Given a predefined set of weights $\tau(w)$, it is straightforward to update the recursive computation of the subsequence kernel so that it reflects the new weighting scheme.

If all the word weights are set to 1, then the new kernel is equivalent to the old one. What we want, however, is a set of weights where words that are correlated with either of the two arguments are given lower weights. For any word, the decrease in weight should reflect the degree of correlation between that word and the two arguments. Before showing the formula used for computing the word weights, we first introduce some notation:

- Let $X \in \mathcal{X}$ be an arbitrary bag, and let $X.a_1$ and $X.a_2$ be the two arguments associated with the bag.
- Let $C(X)$ be the size of the bag (i.e. the number of sentences in the bag), and $C(X, w)$ the number of sentences in the bag X that contain the word w. Let $P(w|X) = C(X, w)/C(X)$.
- Let $P(w|X.a_1 \vee X.a_2)$ be the probability that the word w appears in a sentence due only to the presence of $X.a_1$ or $X.a_2$, assuming $X.a_1$ and $X.a_2$ are independent causes for w.

The word weights are computed as follows:

$$\begin{aligned} \tau(w) &= \frac{C(X,w) - P(w|X.a_1 \vee X.a_2) \cdot C(X)}{C(X,w)} \\ &= 1 - \frac{P(w|X.a_1 \vee X.a_2)}{P(w|X)} \end{aligned} \tag{4.7}$$

The quantity $P(w|X.a_1 \vee X.a_2) \cdot C(X)$ represents the expected number of sentences in which w would occur, if the only causes were $X.a_1$ or $X.a_2$, independent of each other. We want to discard this quantity from the total number of occurrences $C(X,w)$, so that the effect of correlations with $X.a_1$ or $X.a_2$ is eliminated.

We still need to compute $P(w|X.a_1 \vee X.a_2)$. Because in the definition of $P(w|X.a_1 \vee X.a_2)$, the arguments $X.a_1$ and $X.a_2$ were considered independent causes, $P(w|X.a_1 \vee X.a_2)$ can be computed with the *noisy-or* operator (Pearl, 1986):

$$\begin{aligned} P(w|X.a_1 \vee X.a_2) &= 1-(1-P(w|a_1)) \cdot (1-P(w|a_2)) \\ &= P(w|a_1)+P(w|a_2)-P(w|a_1) \cdot P(w|a_2) \end{aligned} \tag{4.8}$$

The quantity $P(w|a)$ represents the probability that the word w appears in a sentence due only to the presence of a, and it could be estimated using counts on a sufficiently large corpus. For the experimental evaluation, we used the following approximation: given an argument a, a set of sentences containing a are extracted from web documents (details in Section 4.7.2). Then $P(w|a)$ is simply approximated with the ratio of the number of sentences containing w over the total number of sentences, i.e. $P(w|a) = C(w,a)/C(a)$. Because this may be an overestimate (w may appear in a sentence containing a due to causes other than a), and also because of data sparsity, the quantity $\tau(w)$ may sometimes result in a negative value – in these cases it is set to 0, which is equivalent to ignoring the word w in all subsequence patterns.

4.7 MIL Experimental Results

4.7.1 Systems

The experimental evaluation was designed to answer the following questions:

- How adequate is the subsequence kernel for relation extraction, as opposed to using a much simpler approach such as a bag-of-words kernel;
- How successful is the word weights schema from Section 4.6.4 at reducing the Type I bias;
- How does the MIL approach compare against the SIL setting.

Consequently, we evaluated the following four systems:

■ **SSK–MIL**: This corresponds to the MIL formulation from Section 4.6.2, with the original subsequence kernel modified as described at the end of the same section.

■ **SSK–T1**: This is the SSK–MIL system augmented with word weights, so that the Type I bias is reduced, as described in Section 4.6.4.

■ **BW-MIL**: This is a bag-of-words kernel, in which the relation examples are classified based on the unordered words contained in the sentence. This baseline shows the performance of a standard text-classification approach to the problem.

■ **SSK–SIL**: This corresponds to the original subsequence kernel trained with traditional, single instance learning (SIL) supervision. For evaluation, we train on the manually labeled instances from the test bags. We use a combination of one positive bag and one negative bag for training, while the other two bags are used for testing. The results are averaged over all four possible combinations. Note that the supervision provided to SSK–SIL requires significantly more annotation effort, therefore, given a sufficient amount of training examples, we expect this system to perform at least as well as its MIL counterpart.

The subsequence kernel version described at the end of Section 4.6.2 was used as a custom kernel in the LibSVM[4] Java package. When run with the default parameters, the results were extremely poor – too much weight was given to the slack term in the objective function. Minimizing the regularization term is essential in order to capture subsequence patterns shared among positive bags. Therefore LibSVM was modified to solve the optimization problem from Figure 4.11, where the capacity parameter C is normalized by the size of the transformed dataset. In this new formulation, C is set to its default value of 1.0 – changing it to other values did not result in significant improvement. The trade-off between false positive and false negative errors is controlled by the parameter c_p. When set to its default value of 0.5, false-negative errors and false positive errors have the same impact on the objective function. As expected, setting c_p to a smaller value (0.1) resulted in better performance. Tests with even lower values did not improve the results.

4.7.2 Datasets

We created two datasets for experimental evaluation: one for corporate acquisitions, as shown in Table 4.4, and one for the person-birthplace relation, with the example pairs from Table 4.5. In both tables, the top part shows the training pairs, while the bottom part shows the test pairs. In order to evaluate the relation extraction performance at the sentence level, we manually annotated all instances from the positive test bags. The last column in Tables 4.4 and 4.5 shows, between parentheses, how many instances from the positive test bags are real positive instances. The corporate acquisition test set has a total of 995 instances, out of which 156 are positive. The person-birthplace test set has a total of 601 instances, and only 45 of them are positive. Extrapolating from the test set distribution, the positive bags in the person-birthplace dataset are significantly sparser in real positive instances

[4]http://www.csie.ntu.edu.tw/~cjlin/libsvm

than the positive bags in the corporate acquisition dataset.

Split	+/−	**Arg** a_1	**Arg** a_2	**Bag size**
Training	+	Google	YouTube	1375
	+	Adobe Systems	Macromedia	622
	+	Viacom	DreamWorks	323
	+	Novartis	Eon Labs	311
	−	Yahoo	Microsoft	163
	−	Pfizer	Teva	247
Testing	+	Pfizer	Rinat Neuroscience	50 (41)
	+	Yahoo	Inktomi	433 (115)
	−	Google	Apple	281
	−	Viacom	NBC	231

Table 4.4: Corporate Acquisition Pairs.

Split	+/−	**Arg** a_1	**Arg** a_2	**Bag size**
Training	+	Franz Kafka	Prague	552
	+	Andre Agassi	Las Vegas	386
	+	Charlie Chaplin	London	292
	+	George Gershwin	New York	260
	−	Luc Besson	New York	74
	−	Wolfgang A. Mozart	Vienna	288
Testing	+	Luc Besson	Paris	126 (6)
	+	Marie Antoinette	Vienna	105 (39)
	−	Charlie Chaplin	Hollywood	266
	−	George Gershwin	London	104

Table 4.5: Person Birthplace Pairs.

Given a pair of arguments (a_1, a_2), the corresponding bag of sentences is created as follows:

■ A query string "$a_1 * * * * * * * a_2$" containing seven wildcard symbols between the two arguments is submitted to Google. The preferences are set to search only for pages written in English, with Safesearch turned on. This type of query will match documents where an occurrence of a_1 is separated from an occurrence of a_2

by at most seven content words. This is an approximation of our actual information need: "return all documents containing a_1 and a_2 in the same sentence".

■ The returned documents (limited by Google to the first 1000) are downloaded, and then the text is extracted using the HTML parser from the Java Swing package. Whenever possible, the appropriate HTML tags (e.g. *BR, DD, P*, etc.) are used as hard end-of-sentence indicators. The text is further segmented into sentences with the OpenNLP[5] package.

■ Sentences that do not contain both arguments a_1 and a_2 are discarded. For every remaining sentence, we find the occurrences of a_1 and a_2 that are closest to each other, and create a relation example by replacing a_1 with $\langle e_1 \rangle$ and a_2 with $\langle e_2 \rangle$. All other occurrences of a_1 and a_2 are replaced with a null token ignored by the subsequence kernel.

The number of sentences in every bag is shown in the last column of Tables 4.4 & 4.5. Because Google also counts pages that are deemed too similar in the first 1000, some of the bags can be relatively small.

As described in Section 4.6.4, the word-argument correlations are modeled through the quantity $P(w|a) = C(w,a)/C(a)$, estimated as the ratio between the number of sentences containing w and a, and the number of sentences containing a. These counts are computed over a bag of sentences containing a, which is created by querying Google for the argument a, and then by processing the results as described above.

4.7.3 Results and Discussion

Experimental results are shown in Figures 4.12 & 4.13, where precision is plotted against recall by varying a threshold on the value of the SVM decision function. To avoid clutter, we show only the graphs for the first three systems. The area under

[5]http://opennlp.sourceforge.net

the precision recall curve for all systems is shown in Table 4.6.

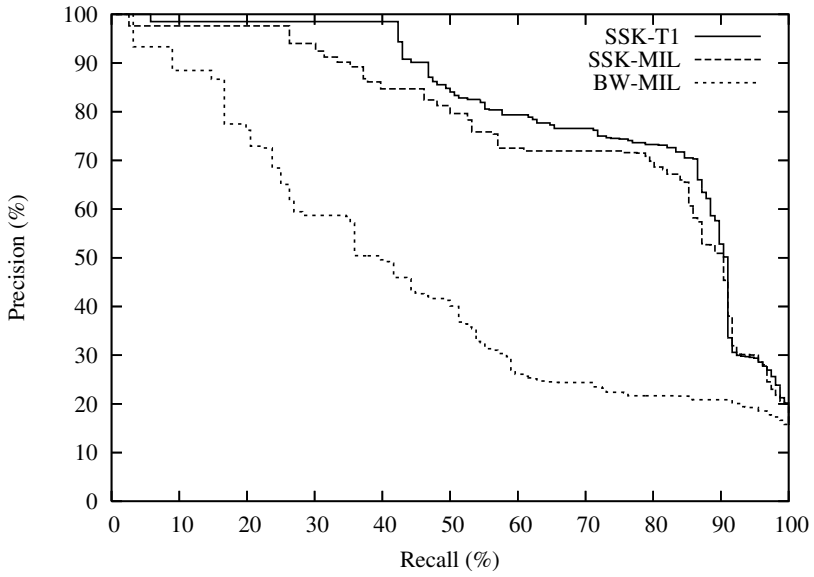


Figure 4.12: Precision-Recall graphs for Corporate Acquisitions dataset.

Dataset	SSK–MIL	SSK–T1	BW–MIL	SSK–SIL
Corporate Acquisitions	76.9%	81.1%	45.9%	80.4%
Person Birthplace	72.5%	78.2%	69.2%	73.4%

Table 4.6: Area Under Precision-Recall Curve.

Overall, the learned relation extractors using the subsequence kernel are able to identify the relationship in novel sentences quite accurately, and significantly outperform a bag-of-words baseline. The BW-MIL approach performs surprisingly well on the Person Birthplace dataset – this might be explained by the relatively small vocabulary that is generally used for expressing the person-birthplace relationship.

The new version of the subsequence kernel SSK–T1 is significantly more accurate in the MIL setting than the original subsequence kernel SSK–MIL. This result empirically demonstrates that the non-uniform word weights approach from

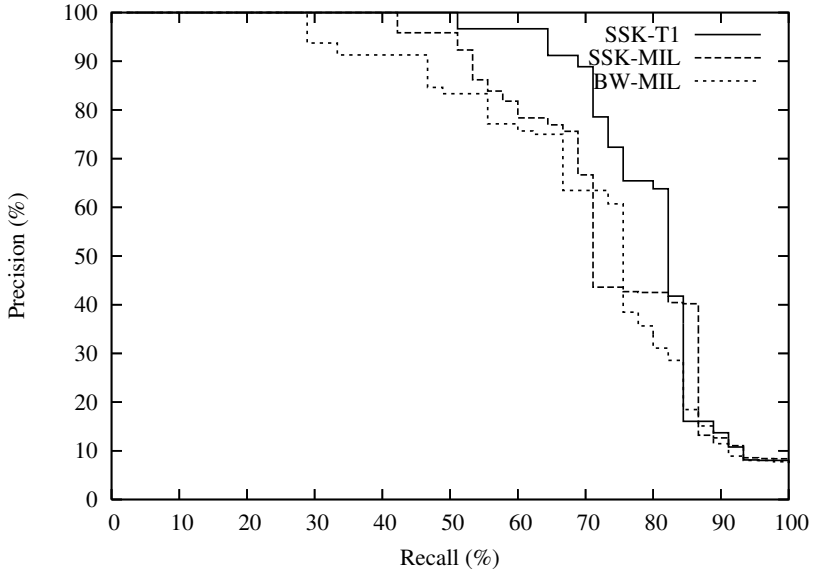


Figure 4.13: Precision-Recall graphs for Person Birthplace dataset.

Section 4.6.4 is effective at reducing the Type I bias. The approach is also general enough to be used in conjunction with other types of kernels, as long as the corresponding features are based on words or word sequences.

The MIL approach SSK–T1 is also competitive with SSK–SIL, which was trained in the SIL setting with a reasonable amount of manually labeled sentence examples. Whereas the only human supervision in the MIL setting consists of 6 pairs of entities (4 positive), the SIL setting is trained on average on 500 manually labeled sentence examples (78 positive) in the Corporate Acquisitions task, and 300 manually labeled sentence examples (22 positive) in the Person Birthplace task.

4.8 Related Work

The development of our two SIL approaches to RE – the subsequence kernel from Section 4.3 and the dependency path kernel from Section 4.4 – can be traced back to

the tree kernels of Zelenko et al. (2003) and Culotta and Sorensen (2004). Zelenko et al. (2003) define a tree kernel over shallow parse representations of text, together with an efficient algorithm for computing it. Experiments on extracting PERSON-AFFILIATION and ORGANIZATION-LOCATION relations from 200 news articles show the advantage of using this new type of tree kernels over three feature-based algorithms. The same kernel was slightly generalized by Culotta and Sorensen (2004) and applied on dependency tree representations of sentences, with dependency trees being created from head-modifier relationships extracted from syntactic parse trees. Experimental results show a clear advantage of the dependency tree kernel over a bag-of-words kernel. However, in a bag-of-words approach the word order is lost. For relation extraction word order is important, and our intuition was that a method that took into account the order between words could achieve a significantly better accuracy, even when dependency information was not available. This was empirically demonstrated by the competitive performance of the subsequence kernel approach, which preserves the word order observed in the training examples.

The performance of the dependency path kernel is highly dependent on the accuracy of the dependency parse of the sentence. For sentences with unreliable syntactic parses, simple word sequences can often provide a good substitute. Consequently, combining dependency information with features based on word sequences could lead to further improvements in performance, as demonstrated by the more recent approaches to relation extraction from (Zhao & Grishman, 2005), (Zhang et al., 2006), and (Jiang & Zhai, 2007).

One of the earliest IE methods designed to work with a reduced amount of supervision is that of Hearst (1992), where a small set of seed patterns is used in a bootstrapping fashion to mine pairs of hypernym-hyponym nouns. Bootstrapping is actually orthogonal to our method, which could be used as the pattern learner in every bootstrapping iteration. A more recent IE system that works by bootstrapping

relation extraction patterns from the web is KnowItAll (Etzioni et al., 2005). For a given target relation, supervision in KnowItAll is provided as a *rule template* containing words that describe the class of the arguments (e.g. "company"), and a small set of seed extraction patterns (e.g. "has acquired"). In the MIL approach from Section 4.6, the type of supervision is different – we ask only for pairs of entities known to exhibit the target relation or not. Also, KnowItAll requires large numbers of search engine queries in order to collect and validate extraction patterns, therefore experiments can take weeks to complete. Comparatively, our MIL approach to relation extraction requires only a small number of queries: one query per relation pair, and one query for each relation argument.

Craven and Kumlien (1999) create a noisy training set for the subcellular-localization relation by mining Medline for sentences that contain tuples extracted from relevant medical databases. To our knowledge, this is the first approach that is using a "weakly" labeled dataset for relation extraction. The resulting bags however are very dense in positive examples, and they are also many and small – consequently, the two types of bias are not likely to have significant impact on their system's performance.

4.9 Chapter Summary

In this chapter we have introduced three learning approaches to relation extraction. Two of them – the subsequence kernel and the dependency kernel – are trained in a single instance learning setting where the supervision is provided as a dataset of labeled sentence examples. In the subsequence kernel approach, each example is represented using sequences of words and word classes anchored at the two entities as implicit features. In the dependency kernel approach, the set of implicit features is based on the shortest path between the two relation arguments in the dependency graph of the sentence. Experiments on biomedical and news corpora demonstrate

that the two methods obtain improved performance when compared with previous relation extraction systems. In the third approach to relation extraction, the only supervision provided is a handful of pairs of entities known to belong or not belong to the desired relationship. The entity pairs are used in conjunction with a very large corpus in order to automatically extract positive and negative bags of sentences. The resulting multiple instance learning approach can sometimes be misled by textual features correlated with the specific entities in the few training pairs provided. Therefore, we have extended the subsequence kernel with a weighting scheme that enables it to focus on features correlated with the target relation rather than with the individual entities. Experiments on extracting two types of relations from web documents show that the new weighting scheme is effective at reducing the number of extraction errors. The overall MIL system is also competitive with its SIL counterpart, which requires significantly more human supervision.

Chapter 5

Future Work

In this chapter we outline potential future work for each of the three basic IE tasks: *named entity recognition*, *named entity disambiguation*, and *relation extraction*. We conclude with a proposal for an integrated approach to IE that aims to solve all subtasks at once, in which improved extraction performance is obtained as a result of exploiting dependencies among the outputs of the three tasks.

5.1 Named Entity Recognition

In Section 4.4 we have argued for the utility of word-word dependencies in the context of relation extraction. A similar argument can also be made in the context of named entity recognition. By abstracting away from the surface structure of text, the graph of word-word dependencies could help in providing more discriminative features for named entity recognition. For example, if MUSICIANS is one of the target named entity categories, then the name *Gil Shaham* shown in the context from Figure 5.1 should be recognized as belonging to this category. However, the actual text of the name is only indicative of the general category of PEOPLE, and words that are very informative such as *play* and *music* are, on the surface, very far from

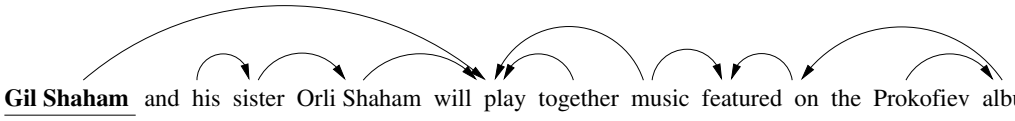


Figure 5.1: Dependency Graph Example.

the name occurrence. The distance between the cue words and the candidate name can be reduced by considering the dependency graph of the sentence, as illustrated in Figure 5.1. Instead of using features based on n-grams around the candidate name, one could use dependency trees of increasing depth rooted at the candidate name, as shown below:

- *Gil Shaham* → *play*
- *Gil Shaham* → *play* → *music*

The verb *play* in the first dependency path is a useful context word for the MUSICIANS category; however, *play* is an ambiguous word that can also refer to other categories, such as SPORTS. The second dependency path helps in establishing the meaning of the verb *play* and thus offers a more discriminative context for disambiguating the name. Words in the dependency tree nodes can also be replaced with word classes such as part-of-speech tags, as done in the two SIL approaches to relation extraction from Chapter 4. Enumerating the large number of resulting features can be avoided by using them only implicitly in a kernel method.

In Chapter 3 we have shown how a dataset of disambiguated named entities can be automatically extracted from Wikipedia articles. Using a similar approach, one could also create a very large training dataset for named entity recognition, thus obviating the need for manually annotated corpora. This can be done as long as the target named entity types can be mapped to categories from the Wikipedia taxonomy. For example, the entity types considered in this thesis – PROTEINS, COMPANIES, PEOPLE, CITIES – all correspond to categories in Wikipedia. A proper

name that appears in a Wikipedia article hyperlinked to its corresponding entity article will be considered as a positive example for a predefined entity type C if the corresponding entity belongs to the category C. However, most entities are linked directly only to low level categories such as American Film Score Composers, or North American Cities. Their membership into higher level categories such as People, or Cities can be decided by navigating the Wikipedia taxonomy graph towards the high level categories. Because there are many types of links, other than `IS-A`, between Wikipedia categories, one needs first to extract an `IS-A` hierarchy from the Wikipedia taxonomy, a task that has been recently addressed by Strube and Ponzetto (2007).

5.2 Named Entity Disambiguation

The same entity can be mentioned multiple times in the same document, using its most popular name (e.g. *Bill Clinton*), alternate names (e.g. *Clinton*), coreferential nouns (e.g. *The President*) or pronouns (e.g. *he*, *him*, *his*). Collectively disambiguating all named entities in a document could lead to further improvements in performance by exploiting soft global constraints such as the *one-sense-per-discourse* effect observed by Gale et al. (1992). Coreference decisions could also help in enriching the context information associated with each name occurrence, as demonstrated in the name discrimination approach of Bagga and Baldwin (1998). As mentioned in Section 3.6, disambiguation accuracy could be further improved by incorporating alternative semantic similarity measures through Latent Semantic Kernels (Cristianini et al., 2001).

Bekkerman and McCallum (2005) have proposed a collective approach to disambiguating web appearances of names that refer to entities known to belong to the same group. For example, the co-authors of a paper form a group, and this information can be used to disambiguate the corresponding name occurrences in

arbitrary web documents. The method could be generalized to work with arbitrary named entities by replacing the social network extracted from paper citations with the network of named entities extracted from each test document. Two named entities that appear in the same sentences would be linked by an edge in the document network. The edges could be associated with predefined relationships, as output by a relation extraction system, or they could be associated simply with the sentence dependency structure anchored at the two names.

5.3 Relation Extraction

The dependency path kernel approach from Section 4.4 is based on the observation that most of the information that is relevant for asserting a relationship between two entity mentions is concentrated in the shortest path between the two mentions in the word-word dependency graph of the sentence. However, sometimes elements that are not on the path, such as negative polarity adverbs, are also essential for the relationship. Our initial solution was to annotate path elements that were linked to negative polarity adverbs. A more principled approach is to augment the shortest dependency path with dependency trees rooted at each content word along the path, and treat them as additional word classes. For example, given the following sentence:

- "**John** wants to travel to **Europe** this summer."

, the corresponding augmented dependency path between *John* and *Europe* is shown in Figure 5.2. To avoid clutter we show only the dependency tree rooted at the word *travel* on the dependency path.

The set of word classes associated with *travel* is extended to contain, besides the original stem and the part-of-speech tags $\{travel, VBD, Verb\}$, all possible dependency trees rooted at *travel*. These will include for instance the paths below:

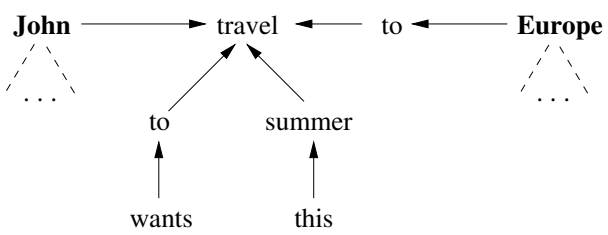


Figure 5.2: Augmented Dependency Path Example.

- *travel* → *to*
- *VBZ* → *to*
- *travel* → *to* → *wants*
- *Verb* → *to* → *wants*

In the kernel approach from Section 4.4, we needed to compute the number of common word classes $c(x_i, y_i)$ between two words x_i and y_i from two dependency paths. For augmented paths, this will require computing the number of common dependency trees rooted at x_i and y_i, which can be done efficiently using a dynamic programming algorithm similar to the convolution tree kernels of Collins and Duffy (2001).

Using augmented dependency paths to create relation extraction features has two main benefits:

1. The resulting model automatically accounts for words that are not on the shortest path but that are still relevant to relation extraction, such as negative polarity adverbs, or control verbs. In the example above, the learned RE system will use the path features containing the control verb *want* in order to decide that *John* is not actually LOCATED in EUROPE at the time of the utterance.

2. The use of rooted dependency trees as additional word classes will result in discriminative features that can generalize better over unseen words.

Notice that the shortest dependency path still lies at the core of the RE system. This approach is highly dependent on the availability of a reliable dependency analysis of the sentence. A more robust approach would combine surface features with dependency based features, a simple solution being to sum up the subsequence kernel from Section 4.3 with the (augmented) dependency kernel from Section 4.4.

Besides named entity disambiguation, another issues that is essential for a successful integration of relation extraction results is that of time. Relationships are time dependent, consequently reasoning with relations should also take time into account. Often the textual assertion of a relation is explicitly qualified with a time expression (e.g. "this summer" in the example from Figure 5.2). We believe that identifying time expressions in text and linking them to the extracted relationships can increase the overall utility of the RE system. Emerging standards for temporal annotation such as TimeML (Boguraev & Ando, 2005) could be used to provide an initial framework for this task.

On the MIL side, we are investigating methods for reducing the Type II bias described in Section 4.6.3, either by modifying the word weights, or by integrating an appropriate measure of word distribution across positive bags directly in the objective function for the MIL problem. Alternatively, implicit negative evidence can be extracted from sentences in positive bags by exploiting the fact that, besides the two relation arguments, a sentence from a positive bag may contain other entity mentions. Any pair of entities different from the relation pair is likely to be a negative example for that relation. For example, the pairs (Google, Yahoo) and (YouTube, Yahoo) from the sentence below can be treated as implicit negative examples:

- **Google** bought **YouTube** not only to keep it away from Yahoo and Microsoft, but also because it offers a service that's superior to its own.

This is similar to the concept of *negative neighborhoods* introduced by Smith and Eisner (2005), and has the potential of reducing both Type I and Type II bias.

5.4 Towards and Integrated IE Model

We have described in Chapter 2 a model that obtains improved extraction performance by taking into account the interdependencies between candidate extractions from the same document. Mutual dependencies can also exist between the outputs of different tasks, such as the three basic IE subtasks: *named entity recognition (NER)*, *named entity disambiguation (NED)*, and *relation extraction (RE)*. Noun phrase *coreference resolution* is another IE task that exhibits mutual interactions with the three tasks mentioned above. Some of these inter-task influences are enumerated below:

- [NER] → [RE]: The entity types of two entity mentions can influence the type of relationship between the two entities. For example, knowing that e_1 is of type PEOPLE and e_2 is of type CITY increases the likelihood that e_1 is LOCATED at e_2. This type of correlations are already captured in the RE models from Chapter 4.

- [RE] → [NER]: Conversely, the relationship between two entities influences the types of entities involved. For example, knowing that e_1 is a COMPANY that has ACQUIRED e_2 increases the likelihood that e_2 is also a COMPANY.

- [NER] → [NED]: NE recognition can reduce the set of potential candidate entities in NE disambiguation. If a name N is recognized as belonging to a category C, and if the name N can denote only two distinct entities e_1 and e_2, with $e_2 \notin C$, then N can be disambiguated only to e_1. This type of influences are implicitly exploited by the NE disambiguation model from Chapter 3.

- [NED] → [RE]: The relation extraction accuracy can be improved by aggregating for each relation instance evidence from multiple documents, as in (Bunescu et al., 2006). It is essential for a successful aggregation that the two argument names be disambiguated for each relation instance.

- [RE] → [NED]: When the category of a name is not sufficient for full disambiguation, the relations in which the denoted entity participates, as inferred from the context of the name occurrence, can be used as additional evidence.

Modeling the mutual influence between NE recognition and relation extraction has been explored by Roth and Yih (2004) with promising results. Modeling the interdependent outputs from all IE subtasks could be achieved either by training one global Markov Network to solve them simultaneously, or by training a separate model for each subtask and then integrate the correlated outputs, as proposed by Punyakanok et al. (2005).

Chapter 6

Conclusions

The research presented in this thesis has focused on the design and evaluation of machine learning models for three important tasks in information extraction: *named entity recognition*, *named entity disambiguation*, and *relation extraction*. The learned extraction models have shown improved extraction performance as a result of their ability to exploit novel useful types of evidence.

We have first described a collective approach to named entity recognition that captures global correlations between candidate extractions through an appropriate formulation using the expressive framework of Relational Markov Networks. The complexity of the resulting graphical model allows only for approximate inference. Motivated by the superior accuracy of exact inference methods, we have also presented a second approach to named entity recognition, cast as phrase based classification with local correlations, in which exact inference can be efficiently achieved in time that is linear in the number of candidate entities. Compared to token classification approaches, our phrase classification models can easily incorporate phrase based features.

The classification of textual occurrences of entity names into predefined categories, as done in named entity recognition, results only in a partial disambiguation

of the names. We have therefore presented an approach to named entity disambiguation that tries to fully disambiguate proper names by linking them to the appropriate entries in Wikipedia, a large online encyclopedia. We have modeled disambiguation as a ranking problem, and showed that improved accuracy is obtained by exploiting learned correlations between context words and categories in the Wikipedia taxonomy.

In the last part of this thesis, we have explored information extraction models that can be trained to mine text documents for predefined relationships between relevant entities. Our approaches to the task of relation extraction differ in the type and the amount of supervision required during training. We have first proposed two relation extraction methods that are trained on documents in which sentences are manually annotated for the required relationships. In the first method, the extraction patterns correspond to sequences of words and word classes anchored at two entity names occurring in the same sentence. These sequences are used as implicit features in a generalized subsequence kernel, with weights that are computed by training in the framework of Support Vector Machines. In the second approach, the implicit extraction features are focused on the shortest path between the two entity names in the word-word dependency graph of the sentence. Finally, in a significant departure from previous learning approaches to relation extraction, we have proposed an extraction model that learns from a much weaker type of supervision. A handful of pairs of entities known to exhibit or not exhibit the desired relationship are used in conjunction with a very large corpus in order to create a training set of bags of sentences, and the subsequence kernel method is extended to handle this special form of supervision. The resulting Multiple Instance Learning approach can sometimes be misled by textual features correlated with the specific entities in the few training pairs provided. Consequently, we have also described a method for weighting features in order to focus on those correlated with the target

relation rather than with the individual entities.

Overall, the research described in this thesis has contributed with learning models that leverage useful new types of evidence in order to obtain improved extraction performance. On a long term scale, we see the proposed methods as a useful step towards building an integrated information extraction model that is robust, accurate, and not overly demanding in terms of human supervision.

Bibliography

Amitay, E., Har'El, N., Sivan, R., & Soffer, A. (2004). Web-a-Where: Geotagging web content. In *Proceedings of 27th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 273–280.

Andrews, S., Tsochantaridis, I., & Hofmann, T. (2003). Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, pp. 561–568 Vancouver, BC. MIT Press.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. ACM Press, New York.

Bagga, A., & Baldwin, B. (1998). Entity-based cross-document coreferencing using the vector space model. In Boitet, C., & Whitelock, P. (Eds.), *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pp. 79–85 San Francisco, California. Morgan Kaufmann Publishers.

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of COLING–ACL '98*, pp. 86–90 San Francisco, CA. Morgan Kaufmann Publishers.

Bekkerman, R., & McCallum, A. (2005). Disambiguating web appearances of people

in a social network. In *Proceedings of WWW-05, the 14th International World Wide Web Conference*.

Berger, A. L., Della Pietra, S. A., & Della Pietra, V. J. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*(1), 39–71.

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

Blaschke, C., & Valencia, A. (2001). Can bibliographic pointers for known biological data be found automatically? Protein interactions as a case study. *Comparative and Functional Genomics*, *2*, 196–206.

Blaschke, C., & Valencia, A. (2002). The frame-based module of the Suiseki information extraction system. *IEEE Intelligent Systems*, *17*, 14–20.

Boguraev, B., & Ando, R. K. (2005). TimeML-compliant text analysis for temporal reasoning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pp. 997–1003 Edinburgh, Scotland, UK.

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

Bunescu, R. C., Mooney, R. J., Ramani, A. K., & Marcotte, E. M. (2006). Integrating co-occurrence statistics with information extraction for robust retrieval of protein interactions from Medline. In *Proceedings of the HLT-NAACL Workshop on Linking Natural Language Processing and Biology (BioNLP'06)*, pp. 49–56 New York, NY.

Bunescu, R., Ge, R., Kate, R. J., Marcotte, E. M., Mooney, R. J., Ramani, A. K., & Wong, Y. W. (2005). Comparative experiments on learning information ex-

tractors for proteins and their interactions. *Artificial Intelligence in Medicine (special issue on Summarization and Information Extraction from Medical Documents)*, *33*(2), 139–155.

Bunescu, R., & Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pp. 9–16 Trento, Italy.

Bunescu, R. C. (2004). Learning for collective information extraction. Tech. rep. TR-05-02, Department of Computer Sciences, University of Texas at Austin. Ph.D. proposal.

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

Bunescu, R. C., & Mooney, R. J. (2005a). A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)*, pp. 724–731 Vancouver, BC.

Bunescu, R. C., & Mooney, R. J. (2005b). Subsequence kernels for relation extraction. In Weiss, Y., Schölkopf, B., & Platt, J. (Eds.), *Advances in Neural Information Processing Systems 18* Vancouver, BC.

Bunescu, R. C., & Mooney, R. J. (2007). Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)* Prague, Czech Republic. To appear.

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

Chieu, H. L., & Ng, H. T. (2003). Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Computational Natural Language Learning (CoNLL-2003)*, pp. 160–163 Edmonton, Canada.

Cimiano, P., Handschuh, S., & Staab, S. (2004). Towards the self-annotating web. In *Proceedings of the 13th international conference on World Wide Web*, pp. 462–471 New York, NY, USA. ACM Press.

Collier, N., Park, H., Ogata, N., Tateisi, Y., Nobata, C., T.Ohta, Sekimizu, T., Imai, H., Ibushi, K., & Tsujii, J. (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)*, pp. 271–272 Bergen.

Collins, M. (1999). *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Collins, M. (2002). Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 489–496 Philadelphia, PA.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. In *Proceedings of Neural Information Processing Systems (NIPS 14)*.

Collins, M. J. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97)*, pp. 16–23.

Cooper, G. (1990). Computational complexity of probabilistic inference using Bayesian belief networks (research note).. *Artificial Intelligence*, *42*, 393–405.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., & Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Springer Verlag, New York, NY.

Craven, M., & Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB-1999)*, pp. 77–86 Heidelberg, Germany.

Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Cristianini, N., Shawe-Taylor, J., & Lodhi, H. (2001). Latent semantic kernels. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML'01)*, pp. 66–73 San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Culotta, A., & Sorensen, J. (2004). Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pp. 423–429 Barcelona, Spain.

Dale, R. (2003). Computational linguistics. *Special Issue on the Web as a Corpus*, *29*(3).

Deerwester, S. C., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*, 391–407.

Della Pietra, S., Della Pietra, V. J., & Lafferty, J. D. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4), 380–393.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

Dietterich, T. G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*(1-2), 31–71.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., & Yates, A. (2005). Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, *165*(1), 91–134.

Fellbaum, C. D. (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Finkel, J. R., Grenager, T., & Manning, C. D. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 363–370 Ann Arbor, MI.

Fleischman, M., Hovy, E., & Echihabi, A. (2003). Offline strategies for online question answering: Answering questions before they are asked. In Hinrichs, E., & Roth, D. (Eds.), *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 1–7.

Franzen, K., Eriksson, G., Olsson, F., Asker, L., Liden, P., & Coster, J. (2002). Protein names and how to find them. *International Journal of Medical Informatics*, *67*(1-3), 49–61.

Frege, G. (1892). Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, *100*, 25–50.

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

Freitag, D., & McCallum, A. (1999). Information extraction with HMMs and shrinkage. In *Papers from the Sixteenth National Conference on Artificial Intelligence (AAAI-99) Workshop on Machine Learning for Information Extraction*, pp. 31–36 Orlando, FL.

Gale, W. A., Church, K. W., & Yarowsky, D. (1992). A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, *26*, 415–439.

Gartner, T., Flach, P., Kowalczyk, A., & Smola, A. (2002). Multi-instance kernels. In *Proceedings of the 19th International Conference on Machine Learning*, pp. 179–186 Sydney, Australia. Morgan Kaufmann.

Ge, X. (2002). *Segmental Semi-Markov Models and Applications to Sequence Analysis*. Ph.D. thesis, University of California, Irvine, Irvine, CA.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721–742.

Gooi, C. H., & Allan, J. (2004). Cross-document coreference on a large scale corpus. In *Proceedings of Human Language Technology Conference / North American Association for Computational Linguistics Annual Meeting* Boston, MA.

Grishman, R. (1995). Message Understanding Conference 6. http://cs.nyu.edu/cs/faculty/grishman/ muc6.html.

Grishman, R. (2003). Information extraction. In Mitkov, R. (Ed.), *Handbook of Computational Linguistics*, pp. 545–559. Oxford University Press.

Hammersley, J., & Clifford, P. (1971). Markov fields on graphs and lattices. Unpublished manuscript.

Hassell, J., Aleman-Meza, B., & Arpinar, I. B. (2006). Ontology-driven automatic entity disambiguation in unstructured text. In *International Semantic Web Conference*, pp. 44–57.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of ACL'92* Nantes, France.

Hockenmaier, J., & Steedman, M. (2002). Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 335–342 Philadelphia, PA.

Hudson, R. (1984). *Word Grammar*. Blackwell.

Jensen, F., Lauritzen, S., & Olesen, K. (1990). Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, *4*, 269–282.

Jiang, J., & Zhai, C. (2007). A systematic exploration of the feature space for relation extraction. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT'07)*, pp. 113–120 Rochester, New York.

Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C. J. C., & Smola, A. J. (Eds.), *Advances in Kernel Methods - Support Vector Learning*, pp. 169–184. MIT Press.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)* Edmonton, Canada.

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

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

Ley, M. (2002). The dblp computer science bibliography: Evolution, research issues, perspectives. In *SPIRE 2002: Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, pp. 1–10 London, UK. Springer-Verlag.

Li, H., Srihari, R. K., Niu, C., & Li, W. (2002). Location normalization for information extraction. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*.

Li, S. Z. (1995). *Markov Random Field Modeling in Computer Vision*. Springer Verlag, New York.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematic Programming*, *45*(3), 503–528.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, *2*, 419–444.

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

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Nahm, U. Y., & Mooney, R. J. (2000). A mutually beneficial integration of data mining and information extraction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pp. 627–632 Austin, TX.

NIST (2000). ACE – Automatic Content Extraction. http://www.nist.gov/speech/tests/ace.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, *29*(3), 241–288.

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

Punyakanok, V., Roth, D., tau Yih, W., & Zimak, D. (2005). Learning and inference over constrained output. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pp. 1124–1129 Edinburgh, Scotland, UK.

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

Ratnaparkhi, A. (1996). A maximum entropy part of speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-96)*, pp. 133–141 Philadelphia, PA.

Ray, S., & Craven, M. (2005). Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of 22nd International Conference on Machine Learning (ICML-2005)*, pp. 697–704 Bonn, Germany.

Remy, M. (2002). Wikipedia: The free encyclopedia. *Online Information Review*, *26*(6), 434. www.wikipedia.org.

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

Santorini, B. (1990). Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). Tech. rep. MS-CIS90-47, Department of Computer and Information Science, University of Pennsylvania.

Sarawagi, S., & Cohen, W. W. (2005). Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17* Vancouver, Canada.

Schutze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, *24*(1), 97–123.

Schwartz, A. S., & Hearst, M. A. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of the 8th Pacific Symposium on Biocomputing*, pp. 451–462 Lihue, HI.

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology Conference / North American Association for Computational Linguistics Annual Meeting (HLT-NAACL-2003)*, pp. 134–141 Edmonton, Canada.

Siolas, G., & d'Alchè Buc, F. (2000). Support vector machines based on a semantic kernel for text categorization. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)*, p. 5205 Washington, DC, USA. IEEE Computer Society.

Smith, N. A., & Eisner, J. (2005). Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL'05*, pp. 354–362 Ann Arbor, Michigan.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA.

Strube, M., & Ponzetto, S. P. (2007). Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-2007)* Vancouver, BC. To Appear.

Sutton, C., & McCallum, A. (2004). Collective segmentation and labeling of distant entities in information extraction. Tech. rep. TR # 04-49, University of Massachusetts. Presented at ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields.

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

Tateisi, Y., Yakushiji, A., Ohta, T., & Tsujii, J. (2005). Syntax annotation for the GENIA corpus. In *Second International Joint Conference on Natural Language Processing (IJCNLP'05)*, pp. 222–227 Jeju Island, Korea.

Tjong Kim Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Computational Natural Language Learning (CoNLL-2003)*, pp. 142–147 Edmonton, Canada.

Vapnik, V. N. (1998). *Statistical Learning Theory*. John Wiley & Sons.

Verspoor, K., Cohen, K. B., Goertzel, B., & Mani, I. (Eds.). (2006). *Proceedings of the HLT-NAACL BioNLP Workshop on Linking Natural Language and Biology*, New York, NY.

Wainwright, M., Jaakkola, T., & Willsky, A. (2001). Tree-based reparameterization framework for approximate estimation on graphs with cycles. In *Proceedings of the Conference on Neural Information Processing Systems*.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2000). Generalized belief propagation. In *Advances in Neural Information Processing Systems 12*, pp. 689–695 Denver, CO.

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

Zhang, M., Zhang, J., Su, J., & Zhou, G. (2006). A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)* Sydney, Australia.

Zhang, Q., Goldman, S. A., Yu, W., & Fritts, J. (2002). Content-based image retrieval using multiple-instance learning. In *Proceedings of 19th International Conference on Machine Learning (ICML-2002)*, pp. 682–689.

Zhao, S., & Grishman, R. (2005). Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 419–426 Ann Arbor, Michigan. Association for Computational Linguistics.

Vita

Razvan Bunescu was born in the Romanian village of Malureni in 1975. After graduating from the Technical & Industrial College in Ramnicu-Valcea in 1993, he went to study Computer Science at the University Politehnica of Bucharest, graduating with an Advanced Studies degree in 1999. He subsequently worked as a software engineer in Paris until 2000 when he joined the NLP group at the Southern Methodist University in Dallas. Two years later, felicitous circumstances brought him to the University of Texas at Austin. The doctoral program in the Department of Computer Sciences at UT has kept him busy with exciting research to this day.

Permanent Address: Department of Computer Sciences
1 University Station C0500
Austin, TX 78712
USA

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.