AUTOMATIC RELATION EXTRACTION AMONG NAMED ENTITIES FROM TEXT CONTENTS

CHEN, JINXIU
(B.Eng. M.Eng., Xiamen University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2006

Acknowledgments

I would like to take this opportunity to thank all the people who helped me to complete this thesis.

I would first like to thank my supervisor, Dr. Donghong Ji, whose insights and guidance have helped me develop this thesis. I greatly appreciate my co-supervisor Dr. Chew Lim Tan, who gave me lots of good advice and invaluable support over the years.

Thanks to my labmates, Xiaofeng Yang, Zhengyu Niu, Jie Zhang, Huaqing Hong, Dan Shen, Juan Xiao etc.. They make the lab a pleasant place to work and have helped me clarify many design and implementation issues through discussions. I would like to thank them for always pushing me to finish my thesis.

Thanks also to my flatmates, Dan Lin, Jin Ben, Xiaofei Qi, Kun Qu and many other friends for making my life in Singapore a wonderful memeory.

Finally, my deepest thanks to my family who provide the love and support I can always count on. To my dad, my mom and my fiance Daiqiang, I love all of you so much!

Table of Contents

Summary

This thesis studies the task of relation extraction, which has received more and more attention in recent years. The task of relation extraction is to identify various semantic relations between named entities from text contents. With the rapid increase of various textual data, relation extraction will play an important role in many areas, such as question answering, ontology construction, and bioinformatics.

The goal of our research is to reduce the manual effort and automate the process of relation extraction. To realize this intention, we investigate semi-supervised learning and unsupervised learning solutions to rival supervised learning methods so that we can resolve the problem of relation extraction with minimal human cost and still achieve comparable performance to supervised learning methods.

First, we present a label propagation (LP) based semi-supervised learning algorithm for relation extraction problem to learn from both labeled and unlabeled data. It represents labeled and unlabeled examples and their distances as the nodes and the weights of edges of a graph, then propagating the label information from any vertex to nearby vertices through weighted edges iteratively, finally inferring the labels of unlabeled examples after the propagation process converges.

Secondly, we introduce an unsupervised learning algorithm based on model order identification for automatic relation extraction. The model order identification

is achieved by resampling-based stability analysis and used to infer the number of relation types between entity pairs automatically.

Thirdly, we further investigate unsupervised learning solution for relation disambiguation using graph based strategy. We define the unsupervised relation disambiguation task for entity mention pairs as a partition of a graph so that entity pairs that are more similar to each other, belong to the same cluster. We apply spectral clustering to resolve the problem, which is a relaxation of such NP-hard discrete graph partitioning problem. It works by calculating eigenvectors of an adjacency graph's Laplacian to recover a submanifold of data from a high dimensionality space and then performing cluster number estimation on such spectral information.

The thesis evaluates the proposed methods for extracting relations among named entities automatically, using the ACE corpus. The experimental results indicate that our methods can overcome the problem of being short of manually labeled relation instances for supervised relation extraction methods. The results show that when only a few labeled examples are available, our LP based relation extraction can achieve better performance than SVM and another bootstrapping method. Moreover, our unsupervised approaches can achieve order identification capabilities and outperform the previous unsupervised methods. The results also suggest that all of the four categories of lexical and syntactic features used in the study are useful for the relation extraction task.

List of Figures

List of Tables

Chapter 1

Introduction

This chapter starts from the motivation of this study. Then it describes the objectives and significance of the thesis and gives an overview of the rest of the thesis.

1.1 Motivation

With the development of the electronic technology, there is a dramatic increase of textual information available in the digital archives and the World Wide Web. However, the structure of these resource is largely concerned with the visual formatting of data and not with the data's syntactic and semantic properties. Hence, within these structured pages exists a vast amount of unstructured text ready to be mined and exploited in technologies like web search, question answering and database generation.

In the face of the huge amounts of resource, how can computers help humans make sense of all this data? Ideally, every piece of information that would ever be needed to answer queries or to sort and search data would be neatly marked in the text with some kind of universally agreed upon standard. However, in practice this

is rarely the case and most data remains a set of words strung together (albeit in a not so arbitrary way).

This ideal was recently popularized by Berners-Lee et al. (2001) in their description of the Semantic Web. In the Semantic Web, meaning and language structure are marked up in addition to page format. The major problem facing the Semantic Web is how to mark up billions and billions of pages. The sheer size of the data makes human annotation infeasible. Furthermore, web designers rarely conform to W3C[1] standards when creating pages. Expecting the designers of tomorrow to add an additional layer of markup in future documents is unrealistic.

One course of action would be to have a computer annotate all this electronic data with the structures that are of interest to humans. This is not trivial. How do we tell or teach a computer to recognizer that a piece of text has a semantic property of interest in order to make correct annotations? This process is called *Information Extraction* (IE).

Information Extraction (IE) is an application of natural language processing that identifies relevant information from text documents in a certain domain and put it in a structural format. Information Extraction is different from the more mature technology of Information Retrieval (IR): IR retrieves relevant documents from collections, while IE extracts relevant information from documents.

Generally, there are two main subtasks in current Information Extraction research, that is, *Entity Extraction* and *Relation Extraction*. In the past decade, a large amount of work has been done and obtained satisfied performance on identifying entities from texts (Bikel et al., 1999; Tjong and De, 2003). Hence, extracting entities is not a focus

[1]World Wide Web Consortium, http://www.w3c.org.

of this thesis. The focus of this thesis will be Relation Extraction, that is, how to teach computers to recognize relationships between entities in unstructured text.

The task of relation extraction was first introduced as part of the Template Element task in MUC 6 (MUC, 1995). Most work at MUC was rule-based, which tried to use syntactic and semantic patterns to capture the corresponding relations by means of manually written linguistic rules. Adaptation for a particular domain entails the collection of knowledge that is needed to operate within that domain. Experience indicates that such collection cannot be undertaken by manual means only, i.e., by enlisting domain experts to provide expertise, and computational linguists to induce the expertise into the system, as the costs would compromise the enterprise. Hence, it is generally agreed that the main barriers to wider use of IE technologies due to the difficulties in adapting systems to new applications and domains. It is also challenging to keep track of dynamic information resources (e.g. web pages).

To address these challenges, recently, there is a trend shift in the research community from knowledge-based approaches to machine learning techniques (McCallum and Jensen, 2003). The application of machine learning techniques to IE attempts to relieve the acquisition bottleneck: turning an IE system into out-of-the-shelf components that can be applied to any domain with ease and require no special expertise in artificial intelligence or computational linguistics.

With the availability of corpora as well as sophisticated NLP tools, recent years have seen the application of machine learning techniques, in the Relation Extraction task (Miller et al., 2000; Zelenko et al., 2002; Culotta and Soresen, 2004; Kambhatla, 2004; Zhou et al., 2005; Brin, 1998; Agichtein and Gravano, 2000; Zhang, 2004; Hasegawa et al., 2004; Zhang et al., 2005). Among them, supervised learning

approaches have received more and more research attention (Miller et al., 2000; Zelenko et al., 2002; Culotta and Soresen, 2004; Kambhatla, 2004; Zhou et al., 2005). However, for supervised learning methods, a large amount of labeled training data is needed, which needs much human labor and time consumption. Hence, the main goal of this study is to automatically extract relations among named entities from text contents with minimal human intervention.

1.2 The Objectives and Significance of this thesis

1.2.1 The Objectives

To overcome the shortcoming of manually labeled data, our research aims to automate the process of relation extraction so that we could reduce the manual effort. To realize this intention, we investigate semi-supervised learning and unsupervised learning resolutions to rival supervised learning methods, so that we could resolve the problem of relation extraction with minimal human cost and still are able to achieve comparable performance with the supervised learning methods.

The first objective is to present a label propagation (LP) based semi-supervised learning approach for the relation extraction task. First, this approach represents labeled and unlabeled examples as vertices of a graph, and then propagates the label information from any vertex to any nearby vertex through weighted edges iteratively. Finally we can infer the labels of unlabeled examples after the propagation process converges. The LP based method overcomes the limitation of local consistency constraint of existing bootstrapping-based semi-supervised learning approaches and performs relation classification based on a global consistency assumption by using the

graph-based method, i.e. LP algorithm.

The second objective is to investigate unsupervised learning method for relation extraction problem with order identification capability. Model order identification is achieved by resampling based stability analysis and used to infer the number of relation types between entity pairs automatically.

The last objective is to introduce a novel application of spectral clustering technique to disambiguate various relations between named entities in a fully unsupervised manner. The spectral clustering based method performs a dimensionality reduction on the context vectors of entity pairs, and provides robustness and efficiency that standard clustering methods do not display in direct use. We would like to verify that the application of spectral clustering algorithm can improve the performance of the above unsupervised relation extraction through experimental evaluation.

1.2.2 The Significance

The greatest significance of this study is that we can use the least annotated training examples to extract relations between entity pairs automatically through semi-supervised and unsupervised manner. Experiments are conducted on the ACE corpus to evaluate the proposed methods. The experimental results show that when only a few labeled examples are available, our Label Propagation based relation extraction can achieve better performance than a Support Vector Machine based supervised method and another bootstrapping method. Regarding the proposed unsupervised approaches, the advantages include: a) it does not need any manual labeling of the relation instances; b) it does not need to pre-define the number of the context clusters or pre-specify the similarity threshold for the clusters. The experimental results show

the effectiveness of our proposed algorithm and improve the performance of relation extraction compared to the previous unsupervised method (Hasegawa et al., 2004).

1.3 Overview of the Thesis

Chapter 2 gives the basic concepts related to relations. It analyzes the properties of relations and describes the task of relation extraction as well as evaluation methods used for this task.

Chapter 3 surveys the previous research work on Relation Extraction. The literature review starts with the Knowledge Engineering approaches, and then concentrates on the machine learning based work, including supervised learning, semi-supervised learning, and unsupervised learning based approaches. Advantages and disadvantages of these approaches are discussed in the chapter.

Chapter 4 gives a brief introduction of the ACE corpus used in our experiments.

Chapter 5 focuses on the knowledge representation of issue of automatic relation extraction task. The chapter first introduces the instance representation for each occurrence of entity pairs, and then describes the feature set adopted in this study.

Chapter 6 presents a graph based algorithm, a label propagation (LP) algorithm, for relation extraction task. It formulates the relation extraction problem in the context of semi-supervised learning, and then provides a detail description of the label propagation algorithm and shows how it works for relation extraction. This chapter also introduces two similarity strategies used in the experiments. In the end of the chapter, analysis and discussion of the experimental results are given.

Chapter 7 describes the design of the unsupervised method for relation disambiguation. The chapter first formulates the unsupervised relation extraction problem,

and then further presents the stability based model analysis algorithm to estimate the "target" number of relation types. This chapter also provides the evaluation method for context clustering result and shows the experimental results for the unsupervised method.

Chapter 8 proposes another improved unsupervised model for relation disambiguation, using a spectral clustering technique. First, the chapter models the unsupervised relation disambiguation problem using the graph based strategy. Second, the chapter presents how to apply the spectral clustering technique to resolve the task, which involves how to transform the clustering space and how the Elongated K-means algorithm works on the space. Finally, we describe experiments and evaluations for the unsupervised method.

Finally, Chapter 9 presents conclusions and suggests future work.

Chapter 2

Background

Relation extraction is the task of detecting and classifying implicit and explicit relations between named entities from text contents. It is a key subproblem of information extraction (IE), and is crucial in many natural language applications, such as question answering (QA), bioinformatics, ontology construction and so on.

This chapter will present the background knowledge about relation and the relation extraction task. The first part of the chapter gives the basic notations and concepts of relation. It analyzes the properties of relation. The second part describes the task of relation extraction and introduces the commonly adopted evaluation methods for this task.

2.1 Relation

2.1.1 What are Relations?

Generally, a relation is defined as a logical or natural association between two or more things; or relevance of one to another; or connection. From the perspective of computational linguistics, *relations capture the association between named entities*. Every relation takes two primary arguments: the two named entities that it links. A named entity is any concept that can be identified in text and is related to other named entities. An entity mention is a reference of to a named entity. Entities may be referenced in a text by their name, indicated by a common noun or noun phrase, or represented by a pronoun. For example, the following are several mentions of a single entity:

> ***Name Mention****: Joe Smith*
>
> ***Nominal Mention****: the guy wearing a blue shirt*
>
> ***Pronoun Mentions****: he, him*

Named entities usually are limited to some entity types. Examples of entity types are person, organization, and location. Here, we give the formal statement of the concepts of named entities and relations:

Definition 2.1 (Named Entity) A named entity can be a single token or a set of consecutive tokens with a predefined boundary. Named entities in a document are labeled as E_1;E_2;... according to their order of appearance, and they take values that range over a set of entity types C^E.

Definition 2.2 (Relation) A (binary) relation $R_{ij} = (E_i; E_j)$ represents the relation

between E_i and E_j, where E_i and E_j are its two arguments. In addition, R_{ij} can range over a set of relation types C^R.

Examples of relations are person-affiliation and organization-location. The person-affiliation relation means that a particular person is affiliated with a certain organization. For instance, the sentence

"John Smith is the chief scientist of the Hardcom Corporation."

conveys the semantic relation "person-affiliation", between the entities "John Smith" (PERSON) and "Hardcom Corporation" (ORGANIZATIONS).

2.1.2 Relation: Explicit / Implicit

Relations that are supported by explicit textual evidence will be distinguished from those that depend on contextual inference on the part of the reader.

We do not include relationships dependent on a reader's knowledge of the world. All relations must be based on textual or contextual evidence found within the scope of the document.

We consider a link to be syntactically explicit when a mention modifies another one, or when two mentions are arguments of the same event. Any link between entities that is implied by the text but not rooted in the syntactic connection between two mentions is Implicit. Implicit relations are understood to be between two entities, while explicit relations are considered to be between mentions of two entities.

2.1.2.1 Explicit Relations

Explicit relations are those for which the document provides explicit textual support. This means that the two entity mentions identified as arguments of the relation occur in one of the following syntactic constructions. These constructions either link one entity to the other as a direct or indirect modifier, or else connect the two entities together as arguments of an event.

◇ Modification

A modification links one entity to the other.

- Copular Predicate Modifier:

  (Eg 2.1) *President Clinton was in Washington today.*

  Relation: Located ("*Clinton*", "*Washington*")

- Prepositional Phrase:

  (Eg 2.2) *The CEO of Microsoft...*

  Relation: Role ("*CEO*", "*Microsoft*")

- Adjectival Modifier/Compound Nominal:

  (Eg 2.3) *The American envoy left the talks early.*

  Relation: Role ("*envoy*", "*American*")

- Possessive:

  (Eg 2.4) *Nathan Myhrvold, Microsoft's chief scientist.*

  Relation: Role ("*Microsoft's chief scientist*", "*Microsoft*")

- Conjoined Phrases and Many-to-one Relationships:

  (Eg 2.5) *the three permanent members of the UN, the US, England, and China*

  Relation: Role ("*the three permanent members of the UN*", "*UN*")

  Role ("*US*", "*the three permanent members of the UN*")

  Role ("*England*", "*the three permanent members of the UN*")

  Role ("*China*", "*the three permanent members of the UN*")

- Formulaic Constructions

  For these standard constructions, we will capture the following relations.

  Reporter sign-off:

  (Eg 2.6) *Jane Clayson, ABC News, South Lake Tahoe.*

  Relation: AT ("*Jane Clayson*", "*South Lake Tahoe*")

  Role ("*Jane Clayson*", "*ABC News*")

  Addresses:

  (Eg 2.7) *Mary Smith, Medford, Mass. I feel we should...*

  Relation: Role ("*Smith*", "*Medford*")

  Elected officials:

  (Eg 2.8) *Senate Majority Leader Trent Lott (R-Miss.)*

  Relation: Role.Member ("*Senate Majority Leader Trent Lott*", "*R*")

  AT.Residence ("*Senate Majority Leader Trent Lott*","*Miss.*")

- Non-Identified Entities as modifiers

In cases where a modifier is not an identified entity, and entity embedded in a modification chain may be promoted.

(Eg 2.9) *Mary Smith at the Paris conference made a statement today.*

Relation: At ("*Smith*", "*Paris*")

In this example, *Paris* modifies *conference*, which in turn PP-modifies *Mary Smith*. Because *conference* is not an identified entity, *Paris* may be promoted through the modification chain to fill the Location argument of the relation. Note that promotion is allowable only through non-identified arguments.

◇ **Events**

The relation was conveyed by the linking both entities to an event.

- Event Clause:

  (Eg 2.10) *At one point, the marchers blocked the main road running through Dura with boulders...*

  Relation: AT ("*the marchers*", "*the main road running through Dura*")

  In Eg 2.10, *the marchers* and *the main road running through Dura* are linked to the blocked event.

  (Eg 2.11) *Adam Merriman of Vail, Colo., who travelled to Japan...*

  Relation: AT ("*Merriman*", "*Japan*")

  In the above case, the arguments are linked through relative clauses.

- Nominalized Event NP:

  (Eg 2.12) *Angry over the release of prisoners in the Irish republic...*

Relation: AT ("*prisoners*", "*the Irish republic*")

2.1.2.2 Implicit Relations

Implicit relations are those relations that are not captured by an explicit relation or a chain of explicit relations but that they believe are conveyed by the document as part of the natural understanding of the document's meaning.

(Eg 2.13) *In what appeared to be effort to divert some flak away from Zhu, Hu Jintao, another member of the Communist Party's all-powerful seven-man Standing Committee, is leading the working committee nominally in charge of devising the streamlining plan.*

In the above example, we can get an implicit relation between Zhu and Standing Committee.

Note that implicit relations should have supporting contextual evidence for the relation and do not include those relations that should be derived by combining an understanding of the document with outside world knowledge. In the following is another example, one article whose dateline was *Copenhagen, Denmark* began with the sentence:

(Eg 2.14) *Prime Minister Poul Rasmussen on Thursday made a surprise announcement of national elections.*

and the remainder of the article all concerned Danish party politics. That document does convey an implicit role relation between Rasmussen and Denmark because the other connections and actions ascribed to Rasmussen in the rest of the article only make sense if we do understand that he is the prime minister of Denmark.

Note that most current research involves explicit relations because of poor inter-annotator agreement in the annotation of implicit relations and their limited number.

2.1.3 Relation vs. Non-relations

From the point of view of computational linguistics, relations that depend on external world knowledge rather than on contextual evidence from the document are regarded as non-relations. For example, transitive conclusions based on relations found in the text do not count as identified relations.

(Eg 2.15) *an Alabama women's clinic*

This example clearly conveys a *Located* explicit relation between the clinic and Alabama, but while it might also suggest through transitivity *Located* relations between the clinic and the South, the US, or the world, such transitive conclusions do not count as markable relations.

2.1.4 Coreference of Relation Mentions

When two relations connect the same two identified entities in exactly the same relationship, they should be coreferenced with the same relation ID. And the values of relation type must be identical. For example:

(Eg 2.16)

ROLE.Member (*"the US"(GPE, E3), "UN"(ORG, E20)*)

ROLE.Member (*"America" (GPE, E3), "the United Nations"(ORG, E20)*)

2.2 Relation Extraction Task

In the introduction chapter, we have mentioned that the problem of information extraction has been roughly divided into two sub-tasks: Entity Extraction and Relation Extraction. The task of Entity Extraction is essentially a classification problem: given a piece of text in a document, the task consists in deciding whether it fits into some entity class. The task of Relation Extraction, also known as event extraction or template filling, additionally aims to establish relations between the classified entities.

(Eg 2.17) *Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results. The Seattle-based company[...].*

Entity Extraction task: identify the entities "Alan Mulally", "Boeing" and "Seattle" as instances of the Classes *PERSON*, *ORGANIZATION*, and *LOCATION* respectively;

Relation Extraction task: identify the relations ""Alan Mulally - Boeing" and "Boeing - Seatle" as instances of the class "*PERSON - AFFILIATION*" and "*ORGANIZATION - LOCATION*".

Entity extraction has received a lot of attention in IE research. Recently, relation extraction is a focal point of attention.

Relation Extraction is the task to detect and classify implicit and explicit relationships between named entities from text contents. It seems clear that extracting such information could improve many applications, such as question answering. Though generally useful, Relation Extraction is still a very complex and difficult issue to be

resolved. And traditional knowledge-based approaches for relation extraction will inevitably face its limitations. Hence, in this thesis we focus on the task of automatic relation extraction problem.

Relation Extraction is an emerging NLP technology, and plays an important role in many applications such as Question Answering (Litkowski, 1999; Katz and Lin, 2003; Jijkoun et al., 2004; Shen and Klakow, 2006), Bioinformatics (Rosario and Hearst, 2004; McDonald et al., 2004a; Huang et al., 2004; McDonald et al., 2005), and Ontology Construction (Navigli and Velardi, 2004; Omelayenko,) and so on.

First of all, relation extraction is a key to question answering. Text documents often hide valuable structured data. For example, a collection of newspaper articles might contain information on the *location* of the headquarters of a number of *organizations*. If we need to find:

What is the location of the headquarters of Microsoft?

we could try and use traditional information retrieval techniques for finding documents that contain the answer to our query (Salton, 1998). The naïve strategy is to find documents in which [LOCATION ⟨ unknown ⟩] and [ORGANIZATION ⟨ Microsoft ⟩] are within each other's vicinity. This strategy can produce nice results, but does not always work. Alternatively, we could answer such a query more precisely if we somehow had available a *table* listing all the organization-location pairs that are mentioned in our document collection. A tuple $\langle o, l \rangle$ in such a table would indicate that the headquarters of organization o are in location l, and that this information was present in a document in our collection. Tuple $\langle Microsoft, Redmond \rangle$ in our table would then provide the answer to our query, Figure 2-1 shows such an example

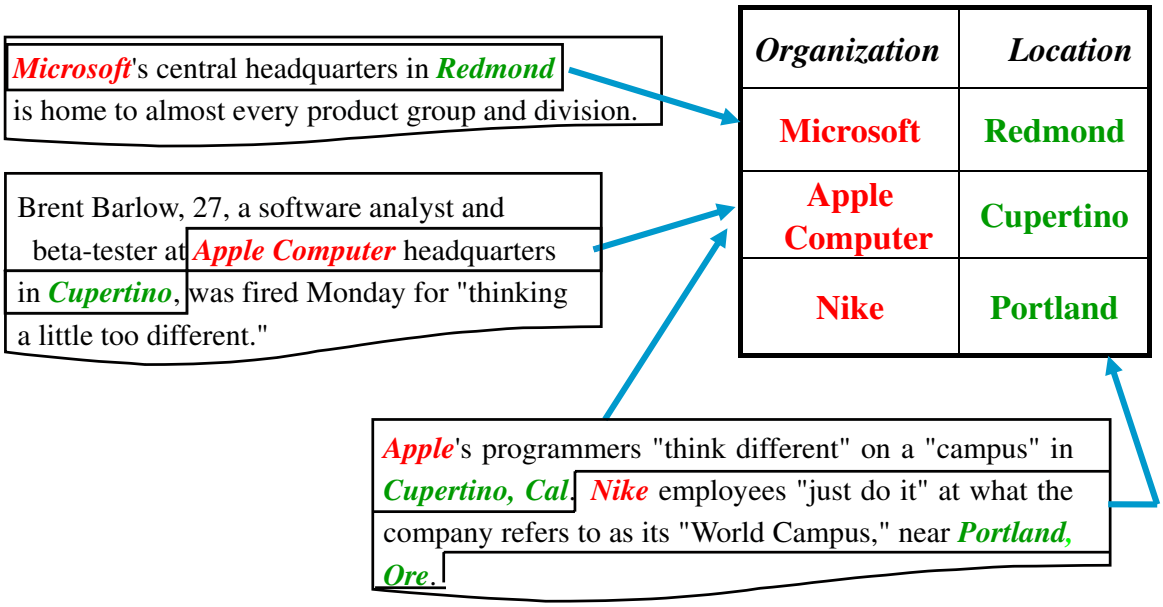


Figure 2-1: An example for tuples of Organization/Location .

for tuples of Organization/Location.

Relation extraction is also very important for bioinformatics. The volume of biological literature is increasing exponentially. This makes it difficult for biologists to keep up with current research or to find particular pieces of information that they need. Using keywords to narrow the search often produces far more candidates than can be properly read (or processed). Therefore, relation extraction techniques have been applied in biomedical domain to identify various relations among biomedical entities, such as DNA, proteins, diseases, etc. Especially, identifying the interactions between proteins is one of the most important challenges in modern genomics, with applications throughout cell biology, including expression analysis, signaling, and rational drug design.

Relation extraction is crucial for ontology construction. With the rapid increase of data on the internet, the process of constructing ontologies manually becomes costly

and difficult for ontology engineering. The researchers of ontology construction can use relation extraction technologies to identify relationships between ontology concepts automatically. This reduces the effort necessary for the knowledge acquisition process.

Due to its importance, relation extraction has received more and more research interest in recent years. In the most recent MUC, relation extraction is defined as an important subtask of information extraction. In the Automatic Content Extraction Program (ACE)[1], which aims to develop automatic content extraction technology to support automatic processing of source languages, the relation extraction task has also been emphasized as an absolutely necessarily objective, ACE RDC subtask.

For a relation extraction task, we would like to answer the following two questions:

Q1 : *Is there a relation between two entities?*

Q2: *If so, which type of relation exists between the two entities?*

The answers to these two questions correspond to the two subtasks. That is,

- Relation Detection
- Relation Classification

2.3 Evaluation of Relation Extraction

The necessity for an evaluation metric for the relation extraction problem started with MUC. The starting points for the development of these metrics were the standard IR metrics of recall and precision. However, the definitions of these measures have been altered from those used in IR, although the names have been retained.

[1]http://www.ldc.upenn.edu/Projects/ACE/

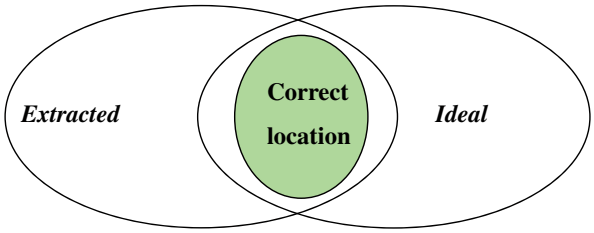


Figure 2-2: The visualization of evaluation metric.

In the relation extraction task, recall may be interpreted as a measure of the fraction of relation instances that has been correctly extracted, and precision as a measure of the fraction of extracted relation instances that is correct. Recall then refers to how many relation instances are correctly extracted, while precision refers to the reliability of the relation instances extracted.

Precision and recall are defined as follows:

$$Precision = \frac{|\,\text{Correct}\,(Extracted \bigcap Ideal)\,|}{|\,Extracted \bigcap Ideal\,|} \tag{2.1}$$

$$Recall = \frac{|\,\text{Correct}\,(Extracted \bigcap Ideal)\,|}{|\,Ideal\,|} \tag{2.2}$$

Both recall and precision are always on the interval [0,1], their optimum being at 1.0. They are, however, inversely related to each other, meaning that by allowing for a lower recall one can achieve a higher precision and vice versa.

And $F - measure$ is the harmonic mean of $Recall$ and $Precision$:

$$F - measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{2.3}$$

Chapter 3

Literature Review for Relation Extraction

Relation extraction has long been recognized as an important and difficult problem by researchers in linguistics, philosophy and computer sciences. This chapter will give a review of literature on the research of relation extraction, which is organized in a way that reflects the trend of the research in this field.

This chapter begins with the traditional knowledge engineering approach and provides a categorization of existing approaches. Then it focuses on presenting the learning based work, which uses supervised learning, semi-supervised learning and unsupervised learning based approaches.

3.1 Knowledge Engineering Approach

In this last decades, to solve the relation extraction problem, many methods have been proposed. In principal, the used approaches can be categorized into two groups:

1. The Knowledge Engineering approach;

2. The Learning approach.

The Knowledge Engineering (KE) approach asks for a system developer, who is familiar with both the requirements of the application domain and the function of the designed IE system. The developer is concerned with the definition of rules used to extract the relevant information. Therefore, a corpus of domain-relevant texts will be available for this task. Furthermore, she or he is free to apply any general knowledge or intuitions in the design of rules. Thus, the performance of the IE system depends on the skill of the knowledge engineer. The KE approach uses an iterative process, whereas within each iteration the rules are modified as a result of the system's output on a training corpus. Thus, the KE approach demands a lot of effort.

The task of relation extraction was first introduced as part of the Template Element task in MUC6 (MUC, 1995). Most works at MUC were rule-based, which are the representative of the KE approach for relation extraction. They tried to use syntactic and semantic patterns to capture the corresponding relations by means of manually written linguistic rules.

Due to the cumbersome manual generation of extraction rules accomplished by knowledge engineers, research has been directed towards automating this task with learning approaches. Learning approaches do not require system expertise. This approach calls only for someone who has enough knowledge about the domain and

the tasks of the system to annotate the texts appropriately. According to the different machine learning strategy adopted, these approaches may be divided into three categories: supervised learning methods, semi-supervised learning methods and unsupervised learning methods.

3.2 Supervised learning methods

Supervised learning methods learn relation patterns using corpora which have been annotated to indicate the information to be extracted. A range of extraction models have been used.

3.2.1 Integrated Parsing

The system proposed by Miller et al. (2000) used an integrated supervised parsing approach. The novelty of their system is to re-annotate natural language parse trees to include relation information at each non-terminal node. Using the re-annotated trees, it is then possible to train a parser (they use the Collins parser (Collins, 1997)) to parse new sentences and extract relation information accordingly.

To build a statistical parsing model which simultaneously recovers syntactic relation and the information extraction information, Miller et al. (2000) used the following steps:

Step 1: annotate training sentences for entities, descriptors, coreference, links, and relation links;

Step 2: train a Collins parser on the Penn treebank (Marcus et al., 1993), and apply it to the new training sentences. Force the parser to produce parses that are

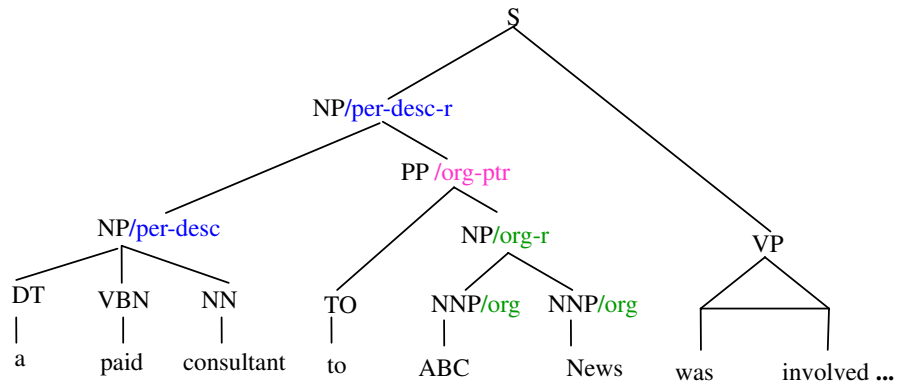


Figure 3-1: An example of a parse tree with entity annotations but no relation annotations.

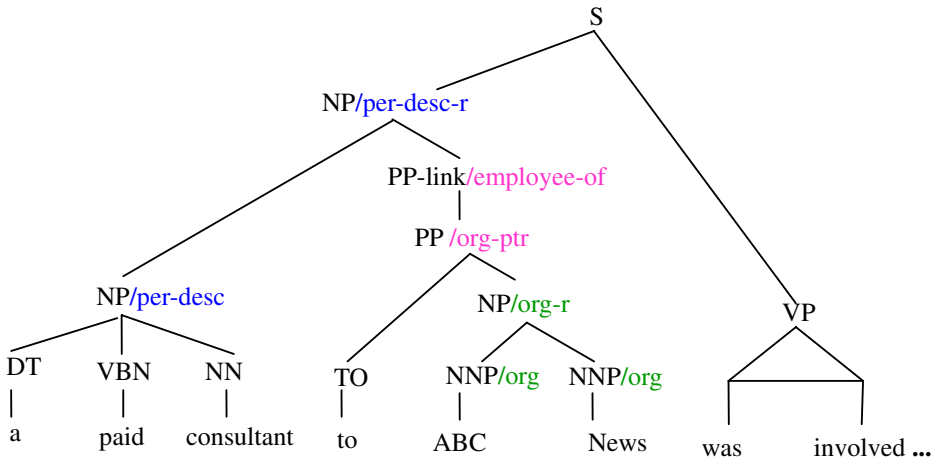


Figure 3-2: An example of an augmented parse tree from Figure 3-1 with relation annotated.

consistent with the entity/descriptor etc. boundaries;

Step 3: augment the parse trees to include the entity and relation information;

Step 4: re-train the Collins parser on the augmented trees in order to tag new sentences.

Miller et al. (2000)'s model is based on a fundamental insight: the realization that by encoding relation and entity information into a parse-tree's non-terminals, results

in the ability to train a state-of-the-art parser to extract relations. No additional models are necessary for relations or entities since they are encoded in the resulting parse tree.

Figure 3-1 shows us an example of an parse tree with entity annotations. In this sentence, the string *a paid consultant to ABC News* is a person description, in which *ABC News* is an organization. Both entities are in an *employee-of* relation. This is the case when the modifier entity is actually part of the entity being modified. In such case, Miller et al. (2000) insert a *link* node directly below the topmost node and the child of that node that subsumes the second entity in the relation (the organization in this case). This node is then labeled with the *employee-of* relation and receives the same syntactic category as the child node. The augmented parse tree with relation annotated can be seen in Figure 3-2.

The above example addressed the case when one entity in the relation modifies the other. When two entities related in a tree are non-overlapping or non-modifying, Miller et al. (2000) handled the case by finding the lowest-most node that subsumes both entities and then the node is augmented to indicate the relation type.

With the augmented syntactic full parse trees with semantic information corresponding to entities and relations, Miller et al. (2000) built generative probability models for the augmented trees. At the training stage, rules for a lexicalized probabilistic context free grammar were estimated that incorporated that semantic attributes. At the evaluation stage, the decoding process yielded a relation-specific interpretation of text, in addition to a syntactic parse.

The system was evaluate on MUC-7, obtained 81% precision and 64% recall in recovering relations.

The intuition behind the integrated parsing approach seems sound. Every entity, relation, POS, and parse tree decision is related and they should all be made at the same time. However, one of the primary disadvantages of the Miller et al. parser is its inability to incorporate long-range features into relation decisions. The reason is that parsing models are constrained to be local (due to complexity issues), that is, Collins parsing model only considers local pairwise dependencies with very little history (relative to the entire tree). Another possible drawback is the use of a generative parse model since generative models cannot easily represent a rich set of dependent features in a computationally tractable manner.

3.2.2 Kernel Methods

3.1.2.1 Zelenko et al. (2002)

Zelenko et al. (2002) designed a model, which extracts relations by computing kernel functions between parse trees, to combat the problems that arose in Miller et al. (2000)'s approach. Unlike Miller et al. (2000)'s work, Zelenko et al. (2002) use shallow parses and not full parses to encode relations. For each shallow parse, the model generates all possible relation instantiations and makes straightforward yes/no classifications on each instantiation to determine what relations, if any, it may contain.

Shallow Parsing A shallow parse is like a full parse, except it only aims to identify the basic surface level components of a sentence, such as noun phrases and entities. The shallow parser used by Zelenko et al. (2002) identifies noun-phases, people, organizations and locations as well as the part-of-speech tags of those words that occur outside noun-phrases or within noun-phrases when there are

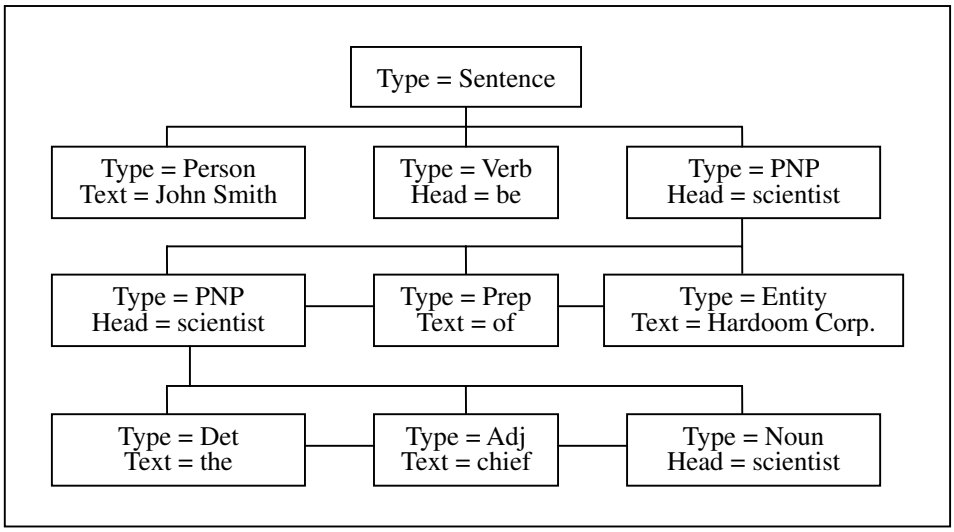


Figure 3-3: An example of input to the system of Zelenko et al. (2002).

non-noun words. Once the shallow parse regions of a sentence have been established, the primary question asked is whether a subtree is an example of the relation of interest. Assuming there is a large set of labeled data, it is possible to create a set of positive and negative examples for classification. For example, say there was interest in the employee-of relation. First a sentence is parsed with the shallow parser. Then for every person/organization pair in the tree, the lowest common node subsuming both entities is found and the subtree rooted at that node extracted. The entity nodes are labeled with a role (e.g., person or organization) in the relation. If those entities are known to be related, then the subtree is given a positive classification and negative otherwise.

Kernels for Relation Extraction Having extracted various positive and negative examples it is fairly straightforward to create a classifier to identify sub-trees containing the relation of interest. Kernel methods do not explicitly generate features. More precisely, an example is no longer a feature vector as it is

common in machine learning algorithms. Instead, examples retain their original representations (of shallow parses) and are used within learning algorithms only via computing a similarity (or kernel) function between them. That is, the approach passes parse tree representations directly into the kernel. Figure 3-3 shows an example of such input. The nodes of the shallow parse trees have attributes, Zelenko et al. (2002) define the following kernel on two subtrees rooted at nodes N_1 and N_2:

$$K(N_1, N_2) = \begin{cases} 0, & \text{if } t(N_1, N_2) = 0; \\ k(N_1, N_2) + K_{child}(N_1, N_2), & \text{otherwise.} \end{cases} \tag{3.1}$$

$$t(N_1, N_2) = \begin{cases} 1, & \text{if } N_1.role = N_2.role \ \& \ N_1.type = N_2.type; \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

$$k(N_1, N_2) = \begin{cases} 1, & \text{if } N_1.text = N_2.text; \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

Classification Every kernel implicitly represents the dot product of the two input examples in some high dimensional space. Therefore, any learning algorithm that can be reformulated so that each input example is only used in dot product calculations with other input examples can be considered a kernel method, since it is always possible to substitute a kernel calculation for a dot product calculation. Zelenko et al. (2002) experiment with both the voted perceptron (Freund and Schapire, 1999) and support vector machines (SVMs). SVMs are similar to the perceptron in that they find a separating hyperplane (when the data is separable), except that SVMs guarantee that the hyperplane returned

Table 3.1: List of features assigned to each node in the dependency tree.

Feature	Example
word	*troops*, *Tikrit*
part-of-speech (*24 values*)	NN, NNP
general-pos (*5 values*)	noun, verb, adj
chunk-tag	NP, VP, ADJP
entity-type	person, geo-political-entity
entity-level	name, nominal, pronoun
Wordnet hypernyms	*social group*, *city*
relation-argument	ARG_A, ARG_B

will be that that which maximizes margin.

3.1.2.2 Culotta and Soresen (2004)

Culotta and Soresen (2004) extended the work of Zelenko et al. (2002) to estimate kernel functions between augmented dependency trees. They represent each relation instance as an augmented dependency tree. A dependency tree represents the grammatical dependencies in a sentence; they augment this tree with features for each node (e.g. part of speech). Table 3.1 lists the features assigned to each node in the dependency tree. Figure 3-4 shows two relation instances, where each node contains the original text plus the features used for the matching. Culotta and Soresen (2004) use the subtree for each pair of entities in a dependency tree that includes both entities instead of the entire tree to reduce noise and emphasize the local characteristics of relations. They choose this representation based on the hypothesis that instances containing similar relations will share similar substructures in their dependency trees.

Culotta and Soresen (2004) evaluate their approach on the ACE corpus, and achieved 63.2 *F-measure* in relation detection and 45.8 *F-measure* in relation classification on the 5 ACE relation types. The kernel in (Culotta and Soresen, 2004) is a recursive

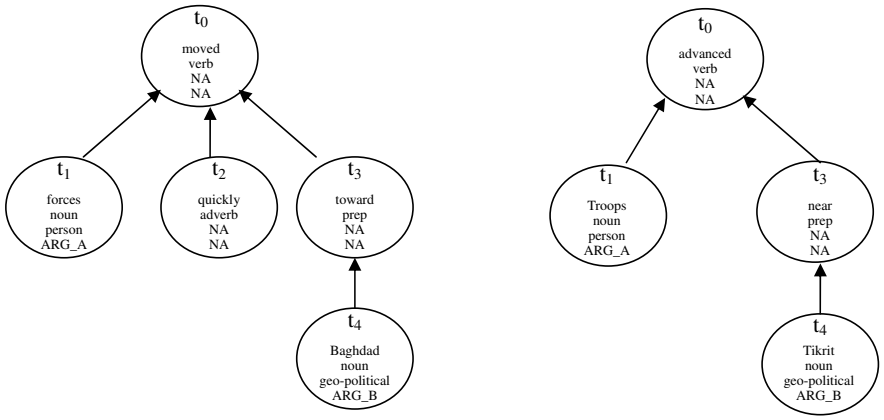


Figure 3-4: Dependency tree for two instances of the *near* relation.

match from the root of a dependency tree down to the leaves where the entity nodes reside, a successful match of two relation examples requires their entity nodes to be at the same depth of the tree. This is a strong constraint on the matching of syntax.

To sum up, kernel-based approaches proposed by Zelenko et al. (2002) and Culotta and Soresen (2004) are able to exploit non-local dependencies since they are not required to model the parse structures of their system (unlike Miller et al. (2000)). This is explicitly handled through the similarity metric. Trees that share more substructure will be given a higher similarity score, making the function global in nature. Furthermore, kernel-based methods are able to explore the implicit feature space without much feature engineering and by reformulating the problem into a yes/no classification problem, they are able to take advantage of state-of-the-art discriminative classification techniques like SVMs (Joachims, 2002) and the voted perceptron (Collins, 2002) to easily handle millions of highly dependent features. And kernel function appears to be a good similarity metric. However, in places the kernel seems a little restrictive. For instance,the indicator function $k(x, y)$ that is only on when substrings match exactly is excessive. A function that takes into account string sim-

ilarity, edit distance or even word overlap might be more indicative. Yet further research work on exploring more feature information is still expected to make it effective with complicated relation extraction tasks.

3.2.3 Feature-based Methods

The most recent model emphasizes feature extraction as proposed by Kambhatla (2004) and Zhou et al. (2005). The feature-based approaches take advantage of discriminative classification techniques to incorporate the diverse lexical, syntactic and semantic information. Unlike kernel based methods, in feature based methods, examples are represented using feature vectors and the discriminative classification model applies directly to predict the type of relation (if any) between every entity mention pairs within each sentence.

Kambhatla (2004) employed maximum entropy models for extracting relations. For each pair of entity mentions, Kambhatla (2004)'s system compute feature streams derived from word, entity type, mention level, overlap, the syntactic parse tree and the dependency tree. All the syntactic features are derived using a statistical parser trained on the Penn TreeBank using the maximum entropy framework (Ratnaparkhi, 1999). The system is evaluated on ACE corpus and achieves 52.8 *F-measure* on the 24 ACE relation subtypes.

Zhou et al. (2005) further explored the feature-based approach with a systematic study on the extensive incorporation of diverse lexical, syntactic and semantic information using SVM. Compared with Kambhatla (2004)'s work, Zhou et al. (2005) separately incorporate the base phrase chunking information, which contributes to most of the performance improvement from the syntactic aspect. They also show

how semantic information like WordNet and name lists can be equipped to further improve the performance. In addition, evaluation on the ACE corpus shows the feature-based approach by Zhou et al. (2005) outperforms tree kernel-based systems (Culotta and Soresen, 2004) by over 20 points in F-measure on 5 ACE relation types.

The above supervised methods have been particularly successful in some specific domains. And we also learned from these methods that the incorporation of diverse features enable systems to combine various kinds of evidence to assist relation extraction. However, the drawback in supervised learning method is that manually tagging of large amounts of training data is time-consuming. Furthermore, it is difficult for one extraction system to be reused across different domains.

3.3 Semi-Supervised Learning methods

Due to the limitation of supervised learning methods, semi-supervised learning methods have been put forward to lessen the corpus annotation requirement. Among the earlier efforts on relation extraction, there are three representative systems that use semi-supervised learning method.

3.3.1 Background: Bootstrapping

Bootstrapping is a general class of semi-supervised learning algorithms. There are two forms of bootstrapping that garner the most attention in the natural language processing community. Blum and Mitchell (1998)'s co-training algorithm and Yarowsky (1995)'s algorithm. At the heart of both algorithms is the notion of a weak learner (or learners) and a large set of unlabeled examples. The algorithm is iterative, using

Table 3.2: Bootstrapping with the Yarowsky's (1995) algorithm. *Conf(D)* is the set of labellings of data D with confidence greater than some threshold.

Algorithm:	Bootstrapping
Input:	A set of seed examples S and a set of unlabeled data D
1.	$T = S$;
2.	Train a classifier C on T;
3.	Label D using C;
4.	$T = Conf(D) \bigcup S$;
5.	Repeat step 2-5 until convergence;
6.	Label D using C.

the output of the learner as training data for the next iteration. Ideally, this process will improve performance. Co-training uses two or more learners, each with a separate view of the unlabeled data. The output of one is then used as the input for others during the next iteration of training. Yarowsky (1995)'s algorithm uses just one trainer, taking the highest confidence examples on each iteration as training for the next iteration. Yarowsky (1995)'s algorithm is the framework primarily deployed by most semi-supervised relation extraction approaches. Table 3.2 outlines the basic Yarowsky (1995)'s algorithm.

When using the Yarowsky (1995)'s algorithm to design system, two considerations must be taken into account, that is, selectivity and coverage. Selectivity refers to our confidence in the classifier's ability to generate precise training examples for future iterations. If the classifier routinely generates false positives, then its accuracy will decrease every iteration, until it becomes of no use whatsoever. This is easily managed by manipulating the classifier to only output positives with extremely high confidence.

However, selectivity must be balanced with coverage. Coverage is the system's

ability to generate new (or all) labeled examples. A classifier that is overly selective will not introduce any new examples and the system will terminate without significantly expanding its seed set. Both these issues play a central role in the considerations of existing system for relation extraction, like (Agichtein and Gravano, 2000).

3.3.2 DIPRE (Brin, 1998)

DIPRE (Dual Iterative Pattern Relation Expansion) by Brin (1998) is a bootstrapping-based system that used a pattern matching system as a classifier to exploit the duality between sets of patterns and relations. The technique was used to extract (*author*, *book*) relation from the World Wide Web.

DIPRE starts with a small set of (*author*, *book*) relations. The system then extracts a tuple for every instance of a (*author*, *book*) seed pair in relative proximity:

[*author*, *book*, *order*, *left*, *middle*, *right*]

where *order* is 1 if the author string occurs before the book string and 0 otherwise, *left/right* are strings containing the 10 characters occurring to the left/right of the match and *middle* the string occurring between the author and book. For example, the tuple extracted for (*Shakespeare, King Lear*) for the string, "*Consider Shakespeare's play King Lear, which tells the tale ...*" would be:

[Shakespeare, King Lear, 1, 'Consider', ''s play', ' which tel']

Each tuple extracted is then grouped by matching *order* and *middle*. For each group of tuples, the longest common suffix of the *left* field and the longest common

prefix of the right field is extracted. Hence, each group induces a pattern:

long-comm-suff(left).AUTHOR.middle.BOOK.long-comm-pref(right)

The above example is for the case when order dictates author before title. Using such a pattern allows the system to extract new examples of (*author*, *book*) pairs. In turn these pairs can generate new patterns.

The primary problem is that some patterns are too easily matched and lead to many false positives. To combat this, DIPRE scores each pattern by |prefix||middle||suffix|, where |s| is the length of string s. Intuitively larger strings are harder to match as they are less common, making these matches more significant. In order to reduce false positives, DIPRE simply throws away all patterns whose score is less than some threshold.

This algorithm is easy to relate to the Yarowsky algorithm. The classifier used by DIPRE is simply a pattern matching system, which is trained by extracting patterns for known (*author*,*book*) pairs. All strings that match at least one of the classifier's patterns are classified as positive and all other strings negative. The (*author*,*book*) pairs in the strings classified as positive and then added to the set of labeled examples to retrain the classifier (i.e., extract more patterns). DIPRE terminates when no new candidate pairs are extracted, or when a human observer decides sufficiently many pairs have been returned.

One of the central insights of DIPRE is that the size of the web allows the use of extremely selective patterns to induce new example pairs of (*author*,*book*). Even with extremely selective patterns, new seed examples will be introduced due to the sheer size of the web. Hence, DIPRE explicitly maintains selectivity by using highly

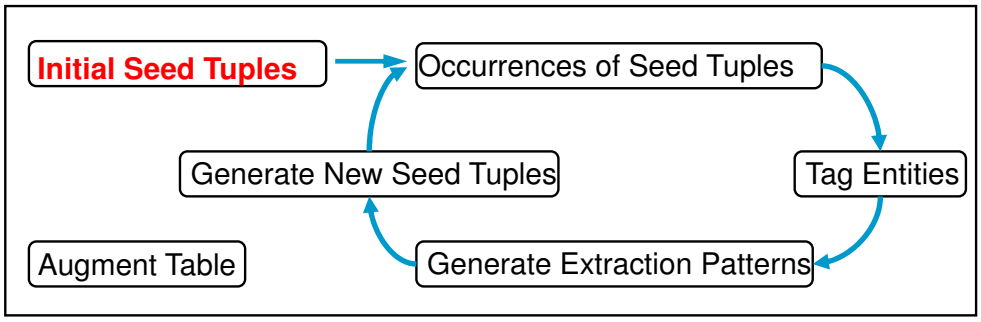


Figure 3-5: The main components of the snowball system.

precise patterns and implicitly increases coverage through the size of the unlabeled data set.

3.3.3 SnowBall (Agichtein and Gravano, 2000)

Snowball by Agichtein and Gravano (2000) is another system that used bootstrapping techniques for extracting relations from unstructured text. Snowball shares much in common with DIPRE, including the employment of the Yarowsky (1995)'s bootstrapping framework as well as the use of pattern matching to extract new candidate relations. The relation that Snowball focuses on is the (organization, location) relation. Figure 3-5 shows the main procedure of Snowball system.

Initial Seed Tuples

We can see that like DIPRE, the Snowball system begins with some initial seed tuples (o_k, l_k). Figure 3-6 gives us some examples of the initial seed tuples.

ORGANIZATION	LOCATION
Microsoft	Redmond
IBM	Armonk
Boeing	Seattle
Intel	Santa Clara

Figure 3-6: The initial seed tuples of snowball.

Occurrences of Seed Tuples and Generate extraction Patterns

Snowball then extracts a tuple for every string in which a known location and organization pair (o_k, l_k) are closed to one another: $[l, e_1, m, e_2, r]$. Where $e_1, e_2 \in \{loc, org\}$ & $e_1 \neq e_2$. m is a feature vector that represents the tokenized terms that occur between the identified pair. Similarly l and r are also feature vectors representing the tokenized terms occurring to the left or right of the pair up to some limit on the number of terms.

Agichtein and Gravano (2000) define a similarity function over extracted tuples:

$$Match(tup_i, tup_j) = \begin{cases} (l_i \cdot l_j) + (m_i \cdot m_j) + (r_i \cdot r_j), & \text{if } e_{1,i} = e_{1,j} \ \& \ e_{2,i} = e_{2,j}; \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

$tup_i = [l_i, e_{1,i}, m_i, e_{2,i}, r_i]$

Clearly tuples that share common terms in their feature vectors are going to have higher similarity over those that do not. Also note that this is a much softer matching criteria than used by DIPRE.

Snowball then induces patterns in two steps. The first step is to cluster all the tuples into a set of groups, $G = \{g_1, ..., g_m\}$, $g_k = \{tup_1^k, ..., tup_n^k\}$, using the similarity

function Match. In the second step, each group, $g_k \in G$ induces a tuple pattern:

$p_{gk} = [l_C, e_1, m_C, e_2, r_C]$

where l_C, m_C and r_C are the centroids of all the left, right and middle feature vectors for the tuples in the group. By the definition of the similarity metric, Match, every tuple belonging to the same group will have identical values for e_1 and e_2.

Snowball handles selectivity by first removing the group that induced the pattern only contained a small number of tuples. A confidence score is then assigned to each pattern:

$$Conf(p_{gk}) = \frac{\text{num}_{\text{pos}}(p_{gk})}{\text{num}_{\text{pos}}(p_{gk}) + \text{num}_{\text{neg}}(p_{gk})} \tag{3.5}$$

where $\text{num}_{\text{pos}}(p_{gk})$ and $\text{num}_{\text{neg}}(p_{gk})$ are the numbers of positive and negative pairs resulting from the application of each pattern p_{gk}, respectively.

Unlike DIPRE, which uses patterns with the highest confidence to introduce new pairs for the next iteration, Snowball uses the confidence measure of patterns to recalculate the confidence of the pairs that the induced patterns extract. Only those pairs with highest confidence are kept for the next iteration.

Tag Entities and generate new Seed Tuples

To extract new pairs, Snowball runs a named-entity tagger over the data to identify all the location and organization entities within the documents. For each organization/location pair, (o, l) that are within the same sentence, the system extracts a tuple, $tup_{(o,l)}$ in the same manner as in the previous section. Hence, a pair that occurs many times will have a set of tuples associated with it, $tup^{(j)}_{(o,l)}$. This tuple is then compared to all the induced patterns that were previously extracted and introduced to the classifier.

For each candidate pair, (o, l), the system records which patterns match the pair with a similarity greater than τ_{sim}, as well as what the similarity value is.

$$M = \{< p_{gk}, Match(tup^j_{(o,l)}, p_{gk}) > | \forall p_{gk}, tup^j_{(o,l)} \, s.t. \, Match(tup^j_{(o,l)}, p_{gk}) > \tau_{sim}\} \tag{3.6}$$

Let $M_i[0]$ be the pattern involved in the i^{th} entry of M and $M_i[1]$ be the similarity score causing this entry.

Snowball defines the confidence of a pair, (o_k, l_k) as:

$$Conf((o_k, l_k)) = 1 - \prod_{i=0}^{|M|}(1 - (Conf(M_i[0]) \cdot M_i[1])) \tag{3.7}$$

The seed set for the next iteration is set to the original seed set, plus the candidate pairs with the highest confidence (confidence greater than τ_{conf}).

One disadvantage of Snowball is that it relies on an intrinsic property of organizations and locations - that every organization has its headquarters in only one location - when calculating the confidence score of a pattern. This property does not hold for all relations. For instance, in the *author-of* relation, one author can be associated with many books and one book with many authors. Even organizations can have multiple headquarters in different parts of the world.

Another disadvantage of Snowball is its reliance on a large number of input parameters for similarity and confidence. The definition of most of these parameters is clear, but there is no guarantee that good values on one set of data will translate to good values on all sets of data. However, these parameters do provide a method for which users can balance their requirements of the system.

Moreover, the use of (Yarowsky, 1995) style bootstrapping algorithms may cause

the problem that the patterns that the system extracts degrade with every iteration since ultimately some errors will be introduced to the system.

3.3.4 Zhang (2004)'s Method

The third system approached the relation classification problem with bootstrapping on top of Support Vector Machines as proposed by Zhang (2004). This system focuses on the ACE subproblem, RDC, and extracts various lexical and syntactic features for the classification task, which includes lexical features, shallow-syntactic features, deep-syntactic features, and so on. However, they don't actually "detect" relations. Rather, their goal is to classify the type of relation between two entities given that they are known to be related.

Table 3.3 shows their bootstrapping procedure based on random feature projection. The basic idea is to generalize the co-training algorithm (Blum and Mitchell, 1998) and relax its two assumptions[1], by only exploiting the potential redundancy in the feature space.

Instead of explicitly "splitting" the feature space, they generate multiple overlapping "views" by random projection from the original feature space. Specifically, in each projection, the features are randomly selected with probability p, and therefore the eventual projected feature space has $p * F$ features on average, where F is the size of the original feature space. Classifiers trained in the projected spaces are then asked to vote on the unlabeled data points. And the agreement measure in Table 3.3

[1]Two assumptions for the original co-training algorithm (Blum and Mitchell, 1998):

- are both sufficient for classification;
- are conditionally independent given the label.

Table 3.3: Zhang (2004)'s bootstrapping procedure based on random feature projection.

Algorithm Bootstrapping using random feature projection

Input: labeled seed set L and unlabeled data set U;
batch size S;
number of projections P;
feature sampling probability p;

Begin

repeat

for $i = 1$ to P **do**

Generate projected feature space F_i, by randomly selecting features probability p;

Project both L and U onto F_i, thus generate L_i and U_i;

Train classifier C_i on L_i;

Run C_i on U_i;

end for

Find (at most) S instances in U with the highest agreement among the P classifiers and assign the most dominant label;

Add them into L;

until No data points available or no reasonable agreement can be reached.

End

are defined using the notion of entropy:

$$H = -\sum_{i}^{C} \frac{\|r_i\|}{B} \log \frac{\|r_i\|}{B} \tag{3.8}$$

The proposed algorithm evaluated on the ACE corpus and showed us that it can reduce the need for labeled training data with sacrificing of performance.

From the survey, we found that current works within the realm of semi-supervised relation extraction mostly use the bootstrapping algorithm. The algorithm does not require a large number of time-consuming hand annotations and one only need pre-define a small set of initial seeds. However, in each iteration step of the bootstrapping procedure, unlabeled examples are classified using a model only trained from labeled data. In other words, it is based on a local consistency assumption: examples close to labeled examples within the same class will have the same labels. This is also the assumption underlying many supervised learning algorithms. Such methods ignore considering the similarity between unlabeled examples and do not perform classification from a global consistency viewpoint, which may fail to exploit appropriate manifold structure in data. Another common feature of these algorithms is that they need to pre-define some initial seeds for any particular relation so that they can bootstrap from the seeds to acquire the relation. However, it is very subjective to determine how to select these seeds and how many seeds to be selected.

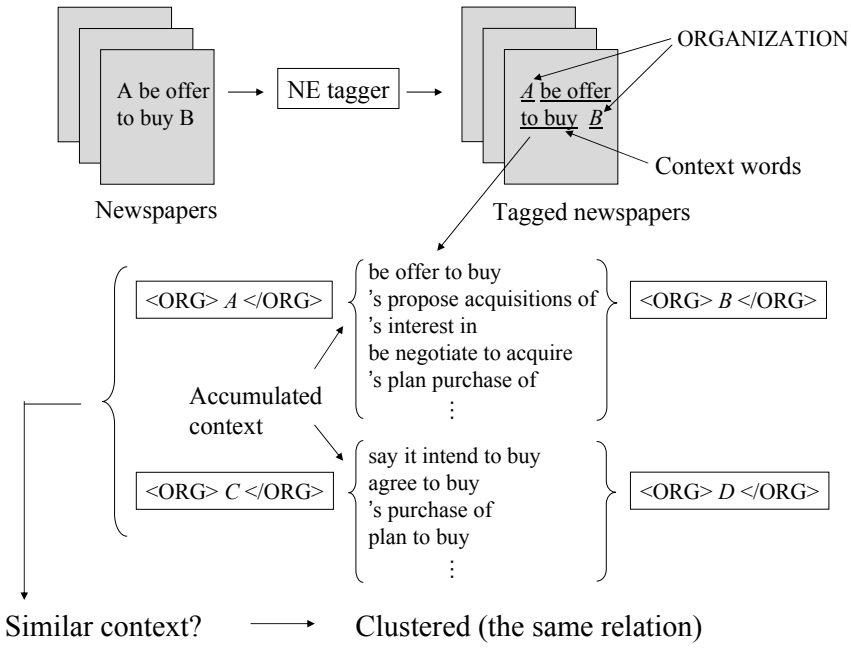


Figure 3-7: The overview of Hasegawa et al. (2004)'s unsupervised system.

3.4 Unsupervised Learning methods

3.4.1 Context Similarity Based: Hasegawa et al. (2004)

Although semi-supervised learning approaches lessen the need of large annotated corpus, they still need some human intervention to pre-define initial seeds. To avoid this constraint, Hasegawa et al. (2004) proposed a method to resolve relation extraction problem in a completely unsupervised way. Figure 3-7 shows the procedure of the unsupervised method. Their assumption is that pairs of entities with same relations between them tend to occur in similar contexts, and the representative words in the contexts can be regarded as a characterization of the relation. Thus, their method contains three key steps:

Collect context vectors: getting co-occurrence pairs of named entities and their

context, where a context vector for each named entity pair consists of the bag of words formed from all intervening words from co-occurrences of two named entities, and each word of a context vector is weighted by $tf * idf$, the product of term frequency and inverse document frequency;

Cluster named entity pairs: clustering the context vectors in which the pairs of entities occur using a hierarchical clustering method, where cosine similarity is adopted to calculate the similarities between the set of contexts of Named Entity pairs;

Label clusters: selecting the most frequent words from the context to label the relation.

Hasegawa et al. (2004)'s system only compare named entity pairs which have the same named entity types, e.g., one *PERSON-GPE* pair and another *PERSON-GPE* pair. Moreover, they assume that for any two particular entities e_1 and e_2, they may hold only one kind of relations. Hence, they accumulate context words for all occurrences of entities e_1 and e_2 to construct the context vector.

For the unsupervised approach, we noticed some limitations. First, they adopted a hierarchical clustering method to cluster the contexts. However, the similarity threshold for the clusters, such as the appropriate number of clusters, is somewhat difficult to pre-define. Second, after context clustering, they select the most frequent words in the contexts to represent the relation that holds between the entities. However, such words may occur frequently in other clusters too, and may not have adequate quality to discriminate between clusters.

3.4.2 Tree based similarity: Zhang et al. (2005)

Zhang et al. (2005) proposed another unsupervised learning method using tree similarity based clustering to extract relations between named entities from a large raw corpus. The method regards relation extraction as a clustering problem on shallow parse trees. The similarity between two relation instances is defined between two parse trees. And clustering entity pairs is based on the similarity score generated by the tree similarity function.

First, Zhang et al. (2005) extend the tree kernels in (Zelenko et al., 2002) to a novel tree similarity measure function. The tree similarity function $K(T_1, T_2)$ over two trees T_1 and T_2, with the root nodes r_1 and r_2, is defined as follows:

$$K(T_1, T_2) = m(r_1, r_2) * \{s(r_1, r_2) + K_c(r_1[c], r_2[c])\} \tag{3.9}$$

where the kernel function $K(T_1, T_2)$ is defined in terms of the similarity function $s(r_1, r_2)$ between the parent nodes, r_1 and r_2, and the similarity function K_c over the two children node sequences $r_1[c]$ and $r_2[c]$.

$$m(p_i, p_j) = \begin{cases} 1, & \text{if } p_i \cdot f_2 = p_j \cdot f_2; \\ 0, & \text{otherwise.} \end{cases} \tag{3.10}$$

$$s(p_i, p_j) = \begin{cases} 1, & \text{if } p_i \cdot f_1 = p_j \cdot f_1 \ \& \ p_i \cdot f_3 = p_j \cdot f_3 \ ; \\ 0.5, & \text{else if } p_i \cdot f_1 = p_j \cdot f_1; \\ 0.25, & \text{if other features match;} \\ 0, & \text{no match.} \end{cases} \tag{3.11}$$

$$K_c(p_1[c], p_2[c]) = \arg \max_{a,b,l(a)=l(b)} \{K(p_1[a], p_2[b])\} \tag{3.12}$$

$$K(p_1[a], p_2[b]) = \sum_{i=1}^{l(a)} K(p_1[a_i], p_2[b_i]) \tag{3.13}$$

Then, following the clustering strategy of Hasegawa et al. (2004), the similarity between parse trees is used in a *hierarchical* clustering algorithm to group entity pairs into different clusters. Finally, each cluster is labeled by an indicative word and unreliable clusters are pruned out.

Since the system by Zhang et al. (2005) mainly follows the hierarchical clustering strategy, it would inevitably face the same limitations in (Hasegawa et al., 2004).

3.5 Summary

In this chapter we gave a literature review of the previous work on relation extraction task. Our discussion focused on machine learning based resolutions. Compared with the knowledge engineering approaches, the machine learning approaches can automatically learn relation patterns from the training data. According to the different machine learning strategies adopted, existing learning based approaches for relation extraction can be divided into three categories: supervised learning methods, semi-supervised learning methods and unsupervised learning methods.

Supervised learning based approaches use a training model to learn relation patterns from annotated data. A range of extraction model have been used. Miller et al. (2000) use an *integrated parsing approach* to extracting relations. In (Miller et al., 2000), full parse trees are used to represent relations and the system makes all relation, entity and syntax decisions at once using a generative probability model.

Later *kernel methods* (Zelenko et al., 2002) have been proposed to combat the problem of Miller et al. (2000)'s system. Unlike Miller et al. (2000), Zelenko et al. (2002) use shallow parses and not full parses to encode relations. The model of (Zelenko et al., 2002) used the output of a shallow parser as its gold standard. For each shallow parse, the model generates all possible relation instantiations and allows for the use of discriminative classification techniques. Culotta and Soresen (2004) extended the work of Zelenko et al. (2002) to estimate kernel functions between augmented dependency trees. Each relation instance is represented as an augmented dependency tree. Recently, *feature-based methods* have been put forward to takes advantage of discriminative classification techniques to incorporate the diverse lexical, syntactic and semantic information for supervised relation extraction task. The major drawback of superivsed learning based methods for relation extraction is that a large amount of labeled training data is needed, which needs quite a lot of human labor and time.

For semi-supervised learning based systems, the primary advantage is that they lessen the corpus annotation requirement. There are three representative systems that use semi-supervised learning method. SnowBall (Agichtein and Gravano, 2000) shares much in common with DIPRE (Brin, 1998), including the employment of the (Yarowsky, 1995)'s bootstrapping framework as well as the use of pattern matching to extract new candidate relations. Zhang (2004) approaches the relation classification problem with bootstrapping on top of support vector machines using various lexical and syntactic features. Most of the current relation extraction based on semi-supervised learning adopted bootstrapping technique. The main problem with Yarowsky (1995) style bootstrapping algorithms according to Yarowsky (1995) is that the patterns that the system extracts degrade with every iteration since ultimately

some errors will be introduced to the system.

To avoid the requirement of manually labeled data for supervised and semi-supervised learning methods, Hasegawa et al. (2004) introduced an unsupervised method for relation discovery from large corpora. Their idea is clustering pairs of named entities according to the similarity of context words intervening between the named entities. Zhang et al. (2005) also introduced an unsupervised learning method to extract relations, which is based on a tree-similarity clustering. This method applied the tree kernel technique in previous supervised method (Zelenko et al., 2002) to calculate the similarity between relation instances, and then cluster them using a hierarchical clustering algorithm as Hasegawa et al. (2004)'s work.

3.6 Comparison with Related Work

The previous semi-supervised method by Zhang (2004) and unsupervised method by Hasegawa et al. (2004) are most related to our works. However, our work in the thesis differs from these others in the following ways:

- Zhang (2004)'s work focuses on the ACE subproblem, RDC and extracts various lexical and syntactic features for the relation classification task. They approach semi-supervised relation classification problem with bootstrapping on top of support vector machines. In contrast, our proposed label propagation based method for relation extraction is a graph based semi-supervised learning method, which can more effectively combine unlabeled data with labeled data in the learning process. To our knowledge, our work is the first one to do relation extraction using graph based semi-supervised learning techniques.

- Zhang (2004)'s semi-supervised work does not actually "detect" relations, but to classify the type of relation between two entities given that they are known to be related. In contract, our proposed LP-based semi-supervised method aims to resolve both relation detection and relation classification problem.

- Hasegawa et al. (2004) propose an unsupervised method for relation discovery. Their assumption is that the same entity pairs in different occurrences have the same relation. In contrast, our proposed unsupervised methods assume that the same entity pairs in different occurrences can have different relation types.

- Hasegawa et al. (2004)'s work adopted a hierarchical clustering method to cluster the contexts. It is somewhat difficult to pre-define the similarity threshold for the clusters, like the appropriate number of clusters. In contrast, our work proposes two resolutions for unsupervised relation extraction, one is resampling based stability clustering (Chapter 7) and the other is based on spectral clustering (Chapter 8). Both of the approaches can achieve model selection.

- Hasegawa et al. (2004)'s work only make use of the context words intervening between the named entities to make the context vectors. In contrast, our work extracts various lexical and syntactic features from the entities and the contexts before, between and after the named entities to construct context vectors.

Chapter 4

Data Set

In this study we evaluate our proposed methods for the task of relation extraction on the official ACE 2003 corpus. It contains 519 files from sources including broadcast news, newswire, and newspaper. According to the scope of the LDC ACE program[1], current research in information extraction has three main objectives: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC). This thesis focuses on the second problem, RDC. The goal of RDC is to detect and characterize relations between EDT entities, for example, that a person is at a location. We dealt with only intra-sentence explicit relations and assumed that all entities have been detected beforehand in the EDT sub-task of ACE. There are five entity types in the ACE corpus, which are:

ORGANIZATION An organization has some formally established association and a persistent, established existence. Typical examples are businesses, government units, sports teams, and formally organized music groups. Industrial sectors are also treated as organizations.

[1] http://www.ldc.upenn.edu/Projects/ACE/

PERSON Each distinct person or set of people mentioned in a document refers to an entity of type person. People may be specified by name ("John Smith"), occupation ("the butcher"), family relation ("dad"), pronoun ("he"), etc., or by some combination of these.

GPE (Geographical-Political Entities) Geo-Political entities are composite entities comprised of a population, a government, a physical location, and a nation (or province, state, country, city, etc.)

LOCATION Location entities are defined on a geographical or astronomical basis which are mentioned in a document and do not constitute a political entity. These include, for example, the solar system, Mars, the continents, the Hudson River, Mt. Everest, and Death Valley.

FACILITY Facilities are artifacts falling under the domains of architecture and civil engineering. For example,buildings and similar facilities designed for human habitation, such as houses, factories, stadiums, office buildings, and so on.

There are also five high-level relations defined in ACE RDC annotation guidelines V3.6. Subtypes will be assigned to every relation further characterizing the identified relationships. For each type, there is a set of possible subtypes. The following lists a categorization of relation types and subtypes, which are:

ROLE affiliation between people and organizations, facilities, and GPEs (Geo-Political Entities). ROLE has six subtypes: Management, General Staff, Member, Owner, Founder, Client, Affiliate-Partner, Citizen-Of, and Other. For example,

(Eg 4.1) *the CEO of Microsoft*

Relation: Role.Management ("*the CEO of Microsoft*", "*Microsoft*")

PART part-whole relationships between organizations, facilities and GPEs. PART has three Subtypes: Part-Of, Subsidiary, and Other.

(Eg 4.2) *Microsoft's headquarters are in Washington.*

Relation: PART.Part-Of ("*Microsoft's headquarters*", "*Microsoft*")

AT location of a Person, Organization, GPE, or Facility entity. For example, a person is at a Location, GPE or Facility if the context indicates that the person was, is or will be there. An Organization is in a Location/GPE if it has a branch there. AT has three Subtypes: Located, Based-In, and Residence.

(Eg 4.3) *The Canadian Hockey Team won in Salt Lake City.*

Relation: AT.Based-In ("*The Canadian Hockey Team*", "*Canada*")

AT.Located ("*The Canadian Hockey Team*","*Salt Lake City*")

NEAR indicates that an entity is explicitly near a location, but not actually in that location or part of that location. Near relations only have one subtype: Relative-Location.

(Eg 4.4) *The city, just west of the mountains,....*

Relation: NEAR.Relative-Location ("*the city*", "*the mountains*")

SOC(Social) personal or professional relationships between people, such as relative, associate, etc. The subtypes for SOC relation include: Parent, Sibling, Spouse, Grandparent, Other-Relative, Other-Personal, Associate, and Other-Professional.

(Eg 4.5) *Joe and Sarah were married 10 years ago.*

Relation: SOC.Spouse (" *Joe*", "*Sarah*")

Table 4.1: Frequency of relation subtypes in the ACE training and devtest corpus.

Type	SubType	Training	Devtest
ROLE	General-Staff	550	149
	Management	677	122
	Citizen-Of	127	24
	Founder	11	5
	Owner	146	15
	Affiliate-Partner	111	15
	Member	460	145
	Client	67	13
	Other	15	7
PART	Part-Of	490	103
	Subsidiary	85	19
	Other	2	1
AT	Located	975	192
	Based-In	187	64
	Residence	154	54
SOC	Other-Professional	195	25
	Other-Personal	60	10
	Parent	68	24
	Spouse	21	4
	Associate	49	7
	Other-Relative	23	10
	Sibling	7	4
	GrandParent	6	1
NEAR	Relative-Location	88	32

Table 4.1 lists the types and subtypes of relations for the ACE Relation Detection and Characterization (RDC) task, along with their frequency of occurrence in the ACE training set and test set.

Chapter 5

Knowledge Representation for Automatic Relation Extraction Models

The knowledge representation problem is a key issue for a learning based approach. Before we introduce our proposed models for automatic relation extraction problem, we would like to first discuss the knowledge representation problem in our models. For example, what kinds of knowledge should be used to indicate the relationship between named entities? And how does one obtain such knowledge?

This chapter will explore the knowledge representation problem for automatic relation extraction problem. We will interpret the instance representation for each occurrence of each entity mention pair and give a description of feature set used in our study.

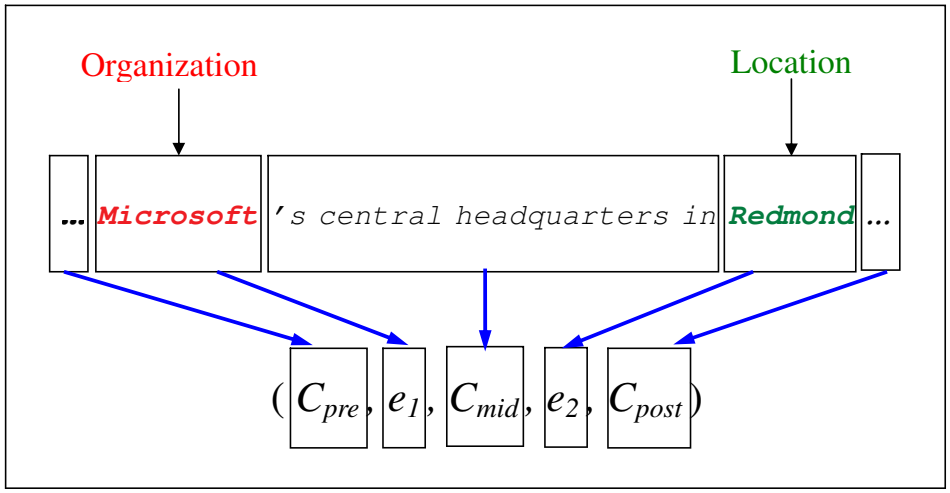


Figure 5-1: An example of relation instance represented by the five-tuple.

5.1 Instance Representation

The problem of relation extraction is to detect and characterize the relationship between named entities. It is assumed that Entity Identification has already taken place beforehand, hence all entity-related information is available at the time of relation extraction. We only deal with intra-sentence explicit relations in this study. In other words, the two entity mentions of the entity arguments of a relation must occur within a common syntactic construction, in this case a sentence.

The basic idea of machine learning based relation extraction is to use machine learning technique to classify and assign an appropriate relation type to an occurrence of two entity pairs in a given context. The information from contexts can help us to capture the characteristics of each occurrence of entity pairs so that we can use the information to discriminate among all entity mention pairs. The first question is how to tell what context an entity mention pair is in. A particularly simple definition is to say that the context of an entity mention pair refers to those context words between

two entity mentions, plus context words before the first entity mention and context words after the second entity mention.

With the definition of context, the problem can be cast into a standard classification framework, which can be formalized as follows:

$$R \rightarrow (C_{pre}, e_1, C_{mid}, e_2, C_{post}) \tag{5.1}$$

where e_1 and e_2 denote the entity mentions, and C_{pre},C_{mid},and C_{post} are the contexts before, between and after the entity mention pairs. A relation label R is assigned to the five-tuple. An example text shown in Figure 5-1.

From Eq.5.1, we can tell that the knowledge for each relation instance comes from two aspects:

- Two entity mentions: e_1 and e_2
- Three contexts: C_{pre},C_{mid},and C_{post}

In the next section we will further explain how to extract features from the two knowledge resources for each instance.

5.2 Feature Inventory

Features are the distinguishing attributes of objects that help to discriminate among them. The choice of features is crucial for the task of automatic relation extraction. In this section we will resolve this question:

- *What features are defined to represent the characteristics of each entity mention pair?*

From the three contexts and the entity mention pair, we extract the lexical and syntactic features, which are computed from the parse trees derived from Charniak Parser (Charniak, 1999) and the Chunklink script[1] written by Sabine Buchholz from Tilburg University. The procedure involves the following steps:

1. Segment the text into sentences using the sentence segmenter provided by the DUC competition[2];
2. Parse the sentence using Charniak Parser (Charniak, 1999);
3. Convert the parse trees into chunklink format using chunklink.pl;
4. Extract and compute features from the chunklink format.

As illustration, Figure 5-2 shows the output information which is produced from the Charniak parser and Chunklink script.

We compute the following lexical and syntactic features to construct the context vectors for each occurrence of an entity mention pair.

Words: Surface tokens of the two entity mentions e_1 and e_2, and words in the three contexts C_{pre},C_{mid},and C_{post}.

Entity Type: the entity type of both entity mentions e_1 and e_2, which can be PERSON, ORGANIZATION, FACILITY, LOCATION and GPE.

POS features: Part-Of-Speech tags corresponding to all tokens in the two entity mentions e_1 and e_2, and words in the three contexts C_{pre},C_{mid},and C_{post}.

[1]Software available at http://ilk.uvt.nl/∼sabine/chunklink/
[2]http://duc.nist.gov/past_duc/duc2003/software/

```
#   arguments: IOB tag: Begin, word numbering: file
#   columns: file_id sent_id word_id iob_inner pos word function heads head_ids iob_chain

1 4 124  B-NP  DT  The  NOFUNC  NOMINAL#11-67#PER_President  125  B-S/B-NP
1 4 125  I-NP  NN  NOMINAL#11-67#PER_President  NP  criticized  128  I-S/I-NP
1 4  126  O  AUX  has  NOFUNC  criticized  128  I-S/B-VP
1 4  127  O  AUX  been  NOFUNC  criticized  128  I-S/I-VP/B-VP
1 4  128  B-VP  VBN  criticized  VP/S  criticized  128  I-S/I-VP/I-VP/B-VP
1 4 129  B-PP  IN  by  PP  criticized  128  I-S/I-VP/I-VP/I-VP/B-PP
1 4 130  B-NP  NNS  NOMINAL#22-95#PER_members  NP  by  129
                                                  I-S/I-VP/I-VP/I-VP/I-PP/B-NP/B-NP
1 4 131  B-PP  IN  of  PP  NOMINAL#22-95#PER_members  130
                                                  I-S/I-VP/I-VP/I-VP/I-PP/I-NP/B-PP
1 4 132  B-NP  JJ  NAME#23-96#ORG_Congress  NOFUNC
         NOMINAL#24-97#ORG_groups  137  I-S/I-VP/I-VP/I-VP/I-PP/I-NP/I-PP/I-NP
1 4 133  I-NP  CC  and  NOFUNC  NOMINAL#24-97#ORG_groups  137
                                                  I-S/I-VP/I-VP/I-VP/I-PP/I-NP/I-PP/I-NP
1 4 134  I-NP  JJ  NAME#14-84#GPE_U.S.  NOFUNC  NOMINAL#24-97#ORG_groups
                                                  137  -S/I-VP/I-VP/I-VP/I-PP/I-NP/I-PP/I-NP
1 4 135  I-NP  JJ  human  NOFUNC  NOMINAL#24-97#ORG_groups  137
                                                  I-S/I-VP/I-VP/I-VP/I-PP/I-NP/I-PP/I-NP
1 4 136  I-NP  NNS  rights  NOFUNC  NOMINAL#24-97#ORG_groups  137
                                                  I-S/I-VP/I-VP/I-VP/I-PP/I-NP/I-PP/I-NP
1 4 137  I-NP  NNS  NOMINAL#24-97#ORG_groups  NP  of  131
                                                  I-S/I-VP/I-VP/I-VP/I-PP/I-NP/I-PP/I-NP
```

Figure 5-2: An example: features derived from the output of the Charniak parser and Chunklink script.

Chunking features: This category of features is extracted from the Chunklink representation, which includes:

- **Chunk tag information** of the two entity mentions e_1 and e_2, and words in the three contexts C_{pre},C_{mid},and C_{post}. The "0" tag means that the word is not in any chunk. The "I-XP" tag means that this word is inside an XP chunk. The "B-XP" by default means that the word is at the beginning of an XP chunk. Here, "XP" can be any chunk, for example, "NP" chunk.
- **Grammatical function** of the two entity mentions e_1 and e_2, and words in the three contexts C_{pre},C_{mid},and C_{post}. The last word in each chunk is its head, and the function of the head is the function of the whole chunk. For example, "NP-SBJ" means a NP chunk as the subject of the sentence. The other words in a chunk that are not the head have "NOFUNC" as their function.
- **IOB-chains** of the heads of the two entity mentions e_1 and e_2. So-called IOB-chain, noting the syntactic categories of all the constituents on the path from the root node to this leaf node of tree.

The position information is also specified in the description of each feature above. For example, word features with position information include:

1) WE1 (WE2): all words in e_1 (e_2)
2) WHE1 (WHE2): head word of e_1 (e_2)
3) WMNULL: no words in C_{mid}
4) WMFL: the only word in C_{mid}

5) WMF, WML: first word, last word in C_{mid} when at least two words in C_{mid}

6) WM2, WM3, ...: second word, third word, ...in C_{mid} when at least three words in C_{mid}

7) WEL1, WEL2, ...: first word, second word, ... before e_1

8) WER1, WER2, ...: first word, second word, ... after e_2

We combine the above lexical and syntactic features with their position information in the contexts to form context vectors. Before that, we filter out low frequency features which appear only once in the dataset.

5.3 Summary

In this chapter we discussed the knowledge representation problem of our automatic relation extraction models. In our study, two aspects of knowledge are used to represent the instance for each occurrences of an entity mention pair, that is, the two entity mentions themselves and the three contexts before, between and after the entity mention pairs. The two types of knowledge reflect the relationships between the two entity mentions, which can explicitly represent the characteristics of each instance for better learning.

The definition of the features is vital for a learning-based system. In this chapter we gave a detailed description of the features used in our study, which include lexical and syntactic features with their position information in the contexts.

With instance representation prepared, in the following three chapters we will have an in-depth discussion of our graph-based semi-supervised model and two other unsupervised based models, respectively. As regards the importance of these differ-

ent features in relation extraction we will provide an analysis in the experimental evaluation in Chapter 8.

Chapter 6

Semi-supervised Relation Extraction with Label Propagation

The advantage of semi-supervised based approaches to relation extraction is that it can reduce the requirement of a large amount of manually annotation corpus for supervised based methods. From the previous literature review, we found that current works on semi-supervised relation extraction solution mostly use the bootstrapping algorithm. However, can such a model accurately represent the relation extraction problem? Or is there another more reasonable learning model?

This chapter will present a novel graph based semi-supervised method for relation extraction. Firstly, we discuss the motivation of the graph based semi-supervised method for relation extraction. Secondly, we formulate the relation extraction problem in the context of semi-supervised learning. Thirdly, we present the solution using Label Propagation based semi-supervised learning. Finally, we show the evaluation result of the proposed method.

6.1 Motivation

As described, to date, most work on semi-supervised learning based methods for relation extraction adopts the bootstrapping algorithm. The bootstrapping algorithm aims to dispense with the need for a large number of time-consuming hand annotations and one only needs to pre-define a small set of initial seeds. It works by iteratively classifying unlabeled examples and adding confidently classified examples into labeled data using a model learned from the augmented labeled data in the previous iteration. However, in each iteration step of bootstrapping procedure, unlabeled examples are classified using a model only trained from the labeled data. The **affinity** among unlabeled examples is not fully explored in this bootstrapping process.

Bootstrapping is based on a **local consistency assumption**: examples close to labeled examples within the same class will have the same labels. This is also the assumption underlying many supervised learning algorithms. Such methods ignore considering the similarity between unlabeled examples and do not perform classification from a global consistency viewpoint, and thus may fail to exploit appropriate manifold structure in data when labeled training data is limited.

To illustrate the consistency assumption, let us consider a toy dataset with a two moon pattern shown in Figure 6-1(a). Every point should be similar to points in its local neighborhood, and furthermore, points in one moon should be more similar to each other than to points in the other moon. The classification results given by the Support Vector Machine and bootstrapping are shown in Figure 6-1(b) and Figure 6-1(c) respectively. According to the assumption of consistency, however, the two moons should be classified as shown in Figure 6-1(d). We can find that both SVM and bootstrapping do not work well and misclassify some points of two moons. The

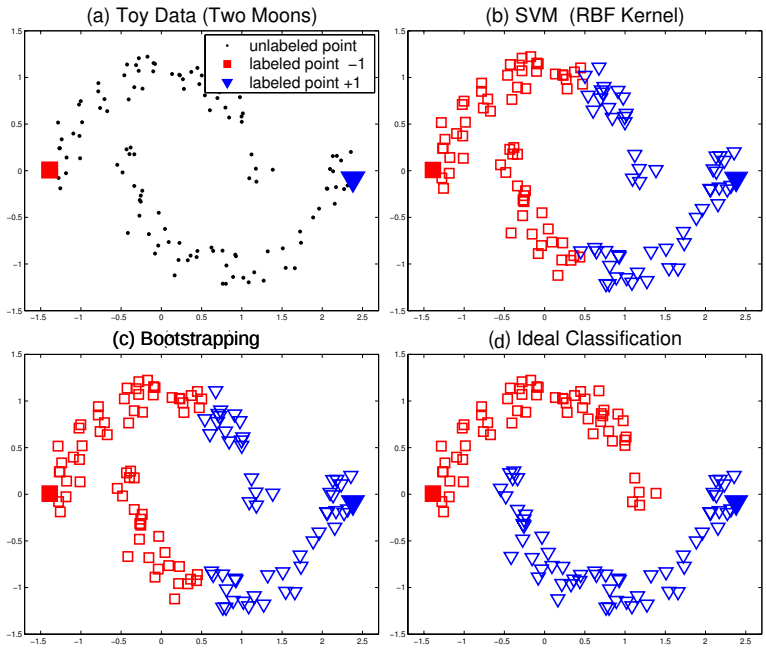


Figure 6-1: Classification result on the two moons pattern dataset. (a) Data set with two labeled points; (b) Classification result given by the SVM; (c) Classification result given by bootstrapping algorithm using k-NN with $k = 1$; (d) Ideal classification.

reason is that both SVM and bootstrapping are based on local consistency assumption and the coherent structure (two moon pattern) in the unlabeled data was not explored when inferring the class boundary.

Recently a promising family of semi-supervised learning algorithm was introduced, which effectively combine unlabeled data with labeled data during the learning process by exploiting manifold structure (cluster structure) in data (Belkin and Niyogi, 2002; Blum and Chawla, 2001; Blum et al., 2004; Zhu and Ghahramani, 2002; Zhu et al., 2003). These graph-based semi-supervised methods usually define a graph where the nodes represent labeled and unlabeled examples in a dataset, and edges (may be weighted) reflect the similarity of examples. These methods usually assume label smoothness over the graph. Graph methods are nonparametric, discriminative, and

transductive in nature. Many graph-based methods can be viewed as estimating a function f on the graph. One wants the labeling function f to satisfy two constraints at the same time:

1. *it should be close to the given labels on the labeled nodes*;
2. *it should be smooth on the whole graph.*

This can be expressed in a regularization framework where the first term is a loss function, and the second term is a regularizer. These methods differ from traditional semi-supervised learning methods in that they use graph structure to smooth the labeling function.

To the best of our knowledge, no work has been done on using graph based semi-supervised learning algorithms for the relation extraction task. Actually the assumption of graph-based methods, that *two points with similar features tend to be in the same class*, fit the problem structure of relation extraction. This observation motivated us to consider how to use graph based methods to detect and identify relations between named entities (Chen et al., 2006a; Chen et al., 2006c). We investigate a label propagation algorithm (LP) (Zhu and Ghahramani, 2002) for relation extraction task. This algorithm works by representing labeled and unlabeled examples as vertices in a connected graph, then propagating the label information from any vertex to nearby vertices through weighted edges iteratively, finally inferring the labels of unlabeled examples after the propagation process converges.

6.2 Modelling semi-supervised relation extraction problem

Like in other learning based applications, before applying a machine learning algorithm to relation extraction, we should first design the learning model of the problem, including:

- *What constitutes an instance of the problem? That is, what is the definition of features to represent each occurrence of entity pairs*?
- *How to construct the graph to represent the knowledge related to the problem*?
- *How to use a graph-based classification to solve the problem*?

For the first question, the solution is to represent each instance using lexical and syntactic features in the three contexts of the entity mention pair and two entity mentions themselves. In Chapter 5, we have given an introduction about what features will be extracted to construct the context vector of an instance.

As regards to the other two questions, they concern how to formulate the relation extraction problem in the context of semi-supervised leaning using the graph based method. We will give answers in the following. First, we show how to construct the graph to represent the knowledge related to the problem.

Let $X = \{x_i\}_{i=1}^{n}$ be a set of instances of all the entity mention pairs, where x_i represents the context vector of the i-th occurrence of entity mention pairs, and n is the total number of occurrences. Let $C = \{r_j\}_{j=1}^{R}$ be a label set, where r_j denotes relation type and R is the total number of relation types.

Then we can construct labeled data and unlabeled data:

- **Labeled data $(x_1, y_1)...(x_l, y_l)$:** The first l examples x_i ($i \leq l$) in X are labeled as y_i ($y_i \in C$), that is, $Y_L = \{y_i\}_{i=1}^{l} \in C$.

- **Unlabeled data $(x_{l+1}, y_{l+1})...(x_{l+u}, y_{l+u})$:** The remaining u ($l+1 \leq u \leq n$) examples are unlabeled, that is, $Y_U = \{y_i\}_{i=l+1}^{l+u}$.

The goal is to predict the label of the unlabeled examples Y_U from X and Y_L.

Intuitively, we assume that:

If two occurrences of entity mention pairs have the similarity context vectors, they tend to hold the same relation type.

Based on the assumption, we define a graph $G = (V, E)$, where the vertices V represent the context vectors of labeled and unlabeled occurrences of entity mention pairs, and the edge E between any two vertices x_i and x_j is weighted so that the closer the vertices not by some distance measure, the larger the weight associated with this edge. Hence, the weights are defined as follows:

$$W_{ij} = exp(-\frac{s_{ij}^2}{\alpha^2}) \tag{6.1}$$

where s_{ij} is the similarity between x_i and x_j calculated by some similarity measures, e.g., cosine similarity, and α is used to scale the weights. In this study, we set α as the average similarity between labeled examples from different classes.

Once the graph is constructed, the next question is how to use a graph-based classification to solve the problem.

To realize the global consistency assumption, the problem of relation extraction can be formulated as a form of propagation on a graph, where a vertex's label propagates to neighboring vertices according to their proximity. Specifically, we need to design a classifying function which is sufficiently smooth with respect to the intrinsic structure revealed by labeled and unlabeled points. In the next section, we will describe a Label Propagation algorithm to construct such a smooth function.

6.3 Resolution

6.3.1 A Label Propagation Algorithm

Given such a graph with labeled and unlabeled vertices, the label propagation algorithm can help us propagate the label information of any vertex in the graph to nearby vertices through weighted edges until a global stable state is achieved. Larger edge weight allows labels to travel through more easily. Thus the closer the examples, more likely they have similar labels (the global consistency assumption).

We define soft label as a vector that is a probabilistic distribution over all the classes. In the label propagation process, the soft label of each initial labeled example is clamped in each iteration to replenish label sources from these labeled data. Thus the labeled data act like sources to push out labels through unlabeled data. With this constant push from labeled examples, the class boundaries will be pushed through edges with large weights and settle in gaps along edges with small weights. If the data structure fits the classification goal, then LP algorithm can use unlabeled data to help learning.

According to the property of classification, we expect that the value of W_{ij} across

different classes is as small as possible and the value of W_{ij} within the same class is as large as possible. This will make label propagation to stay within the same class. This label propagation process will make the labeling function smooth on the graph. In this thesis, we set α as the average similarity between labeled examples from different classes.

Define a $n \times n$ probabilistic transition matrix T

$$T_{ij} = P(j \to i) = \frac{w_{ij}}{\sum_{k=1}^{n} w_{kj}} \tag{6.2}$$

where T_{ij} is the probability to jump from vertex x_j to vertex x_i. We define a $(l+u) \times R$ label matrix Y, where Y_{ij} represents the probabilities of vertex y_i to have the label r_j.

Then the label propagation algorithm consists the following main steps:

Step1 : Initialization

- Set the iteration index $t = 0$;
- Let Y^0 be the initial soft labels attached to each vertex, where $Y_{ij}^0 = 1$ if y_i is label r_j and 0 otherwise.
- Let Y_L^0 be the top l rows of Y^0 and Y_U^0 be the remaining u rows. Y_L^0 is consistent with the labeling in the labeled data and the initialization of Y_U^0 can be arbitrary.

Step 2 : Propagate the labels of any vertex to nearby vertices by $Y^{t+1} = \overline{T}Y^t$, where $\overline{T}$ is the row-normalized matrix of T, i.e. $\overline{T_{ij}} = T_{ij} / \sum_k T_{ik}$, which can maintain the class probability interpretation.

Step 3 : Clamp the labeled data, that is, replace the top l rows of Y^{t+1} with Y_L^0.

Step 4 : Repeat from step 2 until Y converges.

Step 5 : Assign $x_h(l+1 \leq h \leq n)$ with a label: $y_h = \arg\max_j Y_{hj}$.

In the above steps , during each iteration of Step 2, each point receives the information from its neighbors, and also retains its initial information on Step 3. Step 3 is critical. Instead of letting the initially labeled data points 'fade away', we clamp their class distributions, so the probability mass is concentrated on the given class.

6.3.2 Convergence

The above algorithm ensures that the labeled data Y_L never changes since it is clamped in Step 3. Actually we are interested in only Y_U. In the following we show that this algorithm will converge to a unique solution. The transition matrix $\bar{T}$ is split into labeled and unlabeled sub-matrices

$$\bar{T} = \begin{pmatrix} \bar{T}_{LL} & \bar{T}_{LU} \\ \bar{T}_{UL} & \bar{T}_{UU} \end{pmatrix} \tag{6.3}$$

Then it can be shown that the LP algorithm equals to:

$$Y_U = \bar{T}_{UU} Y_U + \bar{T}_{UL} Y_L \tag{6.4}$$

which leads to

$$Y_U = \lim_{n\to\infty} (\bar{T}_{UU})^n Y_U^0 + (\sum_{i=1}^{n} (\bar{T}_{UU})^{(i-1)}) \bar{T}_{UL} Y_L \tag{6.5}$$

where Y_U^0 is the initial value for Y_U. We need to show $(\bar{T}_{UU})^n Y_U^0 \to 0$. Since $\bar{T}$ is row

normalized, and $\bar{T}_{UU}$ is a sub-matrix of $\bar{T}$, it follows

$$\exists \gamma < 1, \sum_{j=1}^{u} (\bar{T}_{UU})_{ij} \leq \gamma, \forall i = 1, ..., u \tag{6.6}$$

Therefore

$$\begin{aligned} \sum_j (\bar{T}_{UU})_{ij}^n &= \sum_j \sum_k (\bar{T}_{UU})_{ik}^{(n-1)} (\bar{T}_{UU})_{kj} \\ &= \sum_k (\bar{T}_{UU})_{ik}^{(n-1)} \sum_j (\bar{T}_{UU})_{kj} \\ &\leq \sum_k (\bar{T}_{UU})_{ik}^{(n-1)} \gamma \\ &\leq \gamma^n \end{aligned}$$

Therefore the row sums of $(\bar{T}_{UU})^n$ converges to zero, which means $(\bar{T}_{UU})^n Y_U^0 \to 0$. Thus the initial value Y_U^0 is not important, since Y_U^0 does not affect the estimation of $\hat{Y}_U$. And obviously,

$$\hat{Y}_U = \lim_{t \to \infty} Y_U^t = (I - \bar{T}_{uu})^{-1} \bar{T}_{ul} Y_L^0 \tag{6.7}$$

is a fixed point (I is $u \times u$ identity matrix). Therefore it is the unique fixed point and the unique solution to the LP iterative algorithm.

As an example, consider the toy problem mentioned earlier, Figure 6-2 shows the classification result of Label propagation algorithm, which shows the convergence process of the algorithm with t increasing from 1 to 400. Note that the initial label information are diffused along the moons. We can find when $t = 400$ the LP algorithm converged to a fixed point, which achieved the ideal classification result.

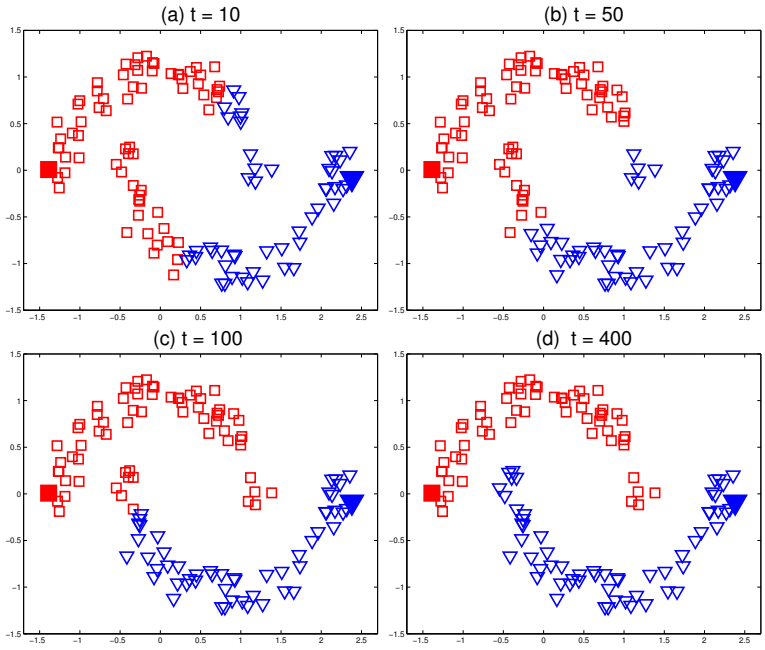


Figure 6-2: Classification result of LP algorithm on two moons pattern dataset. The convergence process of LP algorithm with t varying from 1 to 400 is shown from (a) to (d). Note that the initial label information are diffused along the moons.

6.4 Similarity Measures

The similarity s_{ij} between two occurrences of entity pairs is important to the performance of the LP algorithm. In this chapter, we investigate two similarity measures, cosine similarity measure and Jensen-Shannon (JS) divergence (Lin, 1991). Cosine similarity is a commonly used semantic distance, which measures the angle between two feature vectors α and β.

$$Cosine(\theta) = \frac{\alpha \cdot \beta}{|\alpha||\beta|} \tag{6.8}$$

JS divergence has been used as a distance measure for document clustering, which outperforms cosine similarity based document clustering (Slonim et al., 2002). JS

divergence measures the distance between two probability distributions when feature vectors are considered as probability distribution over features. JS divergence is defined as follows:

$$JS(q,r) = \frac{1}{2}[D_{KL}(q\|\bar{p}) + D_{KL}(r\|\bar{p})] \tag{6.9}$$

$$D_{KL}(q\|\bar{p}) = \sum_{y} q(y)(\log \frac{q(y)}{\bar{p}(y)}) \tag{6.10}$$

$$D_{KL}(r\|\bar{p}) = \sum_{y} r(y)(\log \frac{r(y)}{\bar{p}(y)}) \tag{6.11}$$

$$\bar{p} = \frac{1}{2}(q+r) \tag{6.12}$$

where $JS(q,r)$ represents JS divergence between probability distributions q(y) and r(y) (y is a random variable), which is defined in terms of KL-divergence.

6.5 Experiments and Results

6.5.1 Experiment Setup

We evaluate the above graph based semi-supervised method for relation subtype detection and characterization task on the ACE corpus. Table 4.1 lists the types and subtypes of relations, along with their frequency of occurrence in the ACE training set and test set. We constructed labeled data by randomly sampling some examples from ACE training data and additionally sampling examples with the same size from the pool of unrelated entity pairs for the "NONE" class. We used the remaining examples in the ACE training set and the whole ACE test set as unlabeled data. The testing set was used for final evaluation. And we will evaluate along two subtasks

of relation extraction: Relation Detection and Relation Classification. For Relation Detection, it means that if an entity mention pair is classified not to "NONE" class but to the other 24 subtype classes, then it has a relationship between them.

In this study, to construct the context vector for each instance, we set the mid-context window as the words between the two entity mentions and the pre- and post-context as up to two words before and after the corresponding entity mention.

6.5.2 Experimental Evaluation

In this experiment, to verify the effectiveness of the label propagation algorithm for relation detection and classification, we compare the LP based method with SVM based and bootstrapping based method for relation extraction. For SVM, the labeled data by sampling is used as the training data for SVM model.

LP vs. SVM

Support Vector Machines (SVMs) are a supervised machine learning technique motivated by the statistical learning theory (Vapnik, 1998). Based on the structural risk minimization of the statistical learning theory, SVMs seek an optimal separating hyper-plane to divide the training examples into multi classes and make decisions based on support vectors which are selected as the only effective instances in the training set.

The reason why we choose SVM for this purpose is that it represents the state-of-the-art in the machine learning community and had shown its capability for the supervised relation extraction task (Zhou et al., 2005). Hence, we choose SVM to explore the effectiveness of our semi-supervised learning compared to supervised learning. In

this experiment, we use the LIBSVM tool[1], which is an integrated software for support vector classification, regression, and distribution estimation (one-class SVM). It supports multi-class classfication, and provides probability estimates as well. Moreover, we only apply the simple linear kernel, although other kernels can perform better.

For comparison between SVM and LP, we ran SVM and LP with different sizes of labeled data and evaluate their performance on unlabeled data using *Precision*, *Recall* and *F-measure*. Firstly, we ran SVM or LP algorithm to detect possible relations from unlabeled data. If an entity mention pair is classified not to the "NONE" class but to the other 24 subtype classes, then it has a relation. Then construct labeled datasets with different sampling set size l, including $1\% \times N_{train}$, $10\% \times N_{train}$, $25\% \times N_{train}$, $50\% \times N_{train}$, $75\% \times N_{train}$, $100\% \times N_{train}$ (N_{train} is the number of examples in the ACE training set). If any relation subtype was absent from the sampled labeled set, we redid the sampling. For each size, we performed 20 trials and calculated average scores on the test set over these 20 random trials.

Table 6.1 reports the performance of SVM and LP with different sizes of labled data for relation detection task. We used the same sampled labeled data in LP as the training data for SVM model.

From Table 6.1, we see that both LP_{Cosine} and LP_{JS} achieve higher *Recall* than SVM. Specifically, with small labeled dataset (percentage of labeled data $\leq 25\%$), the performance improvement by LP is significant. When the percentage of labeled data increases from 50% to 100%, LP_{Cosine} is still comparable to SVM in *F-measure* while LP_{JS} achieves slightly better *F-measure* than SVM. On the other hand, LP_{JS} consistently outperforms LP_{Cosine}.

[1] $LIBSVM$: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Table 6.1: The performance of SVM and LP algorithm with different sizes of labeled data for relation detection on relation subtypes. The LP algorithm is run with two similarity measures: Cosine similarity and JS divergence.

	SVM			LP_{Cosine}			LP_{JS}		
Percentage	P	R	F	P	R	F	P	R	F
1%	35.9	32.6	34.4	58.3	56.1	57.1	58.5	58.7	58.5
10%	51.3	41.5	45.9	64.5	57.5	60.7	64.6	62.0	63.2
25%	67.1	52.9	59.1	68.7	59.0	63.4	68.9	63.7	66.1
50%	74.0	57.8	64.9	69.9	61.8	65.6	70.1	64.1	66.9
75%	77.6	59.4	67.2	71.8	63.4	67.3	72.4	64.8	68.3
100%	79.8	62.9	70.3	73.9	66.9	70.2	74.2	68.2	71.1

Table 6.2: The performance of SVM and LP algorithm with different sizes of labeled data for relation detection and classification on relation subtypes. The LP algorithm is run with two similarity measures: cosine similarity and JS divergence.

	SVM			LP_{Cosine}			LP_{JS}		
Percentage	P	R	F	P	R	F	P	R	F
1%	31.6	26.1	28.6	39.6	37.5	38.5	40.1	38.0	39.0
10%	39.1	32.7	35.6	45.9	39.6	42.5	46.2	41.6	43.7
25%	49.8	35.0	41.1	51.0	44.5	47.3	52.3	46.0	48.9
50%	52.5	41.3	46.2	54.1	48.6	51.2	54.9	50.8	52.7
75%	58.7	46.7	52.0	56.0	52.0	53.9	56.1	52.6	54.3
100%	60.8	48.9	54.2	56.2	52.3	54.1	56.3	52.9	54.6

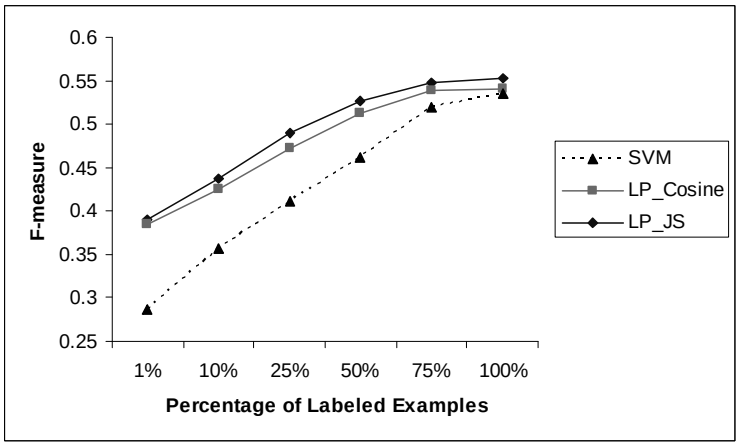


Figure 6-3: Comparison of the performance of SVM and LP with different sizes of labeled data for Relation Classification

Table 6.2 reports the performance of relation classification by using SVM and LP with different sizes of labled data. And the performance describes the average values of *Precision*, *Recall* and *F-measure* over major relation subtypes.

From Table 6.2, we see that LP_{Cosine} and LP_{JS} outperform SVM by *F-measure* in almost all settings of labeled data, which is due to the increase of *Recall*. With smaller labeled dataset (percentage of labeled data $\leq 50\%$), the gap between LP and SVM is larger. When the percentage of labeled data increases from 75% to 100%, the performance of LP algorithm is still comparable to SVM. On the other hand, the LP algorithm based on JS divergence consistently outperforms the LP algorithm based on Cosine similarity. Figure 6-3 visualizes the accuracy of the three algorithms.

As shown in Figure 6-3, the gap between SVM curve and LP_{JS} curves is large when the percentage of labeled data is relatively low.

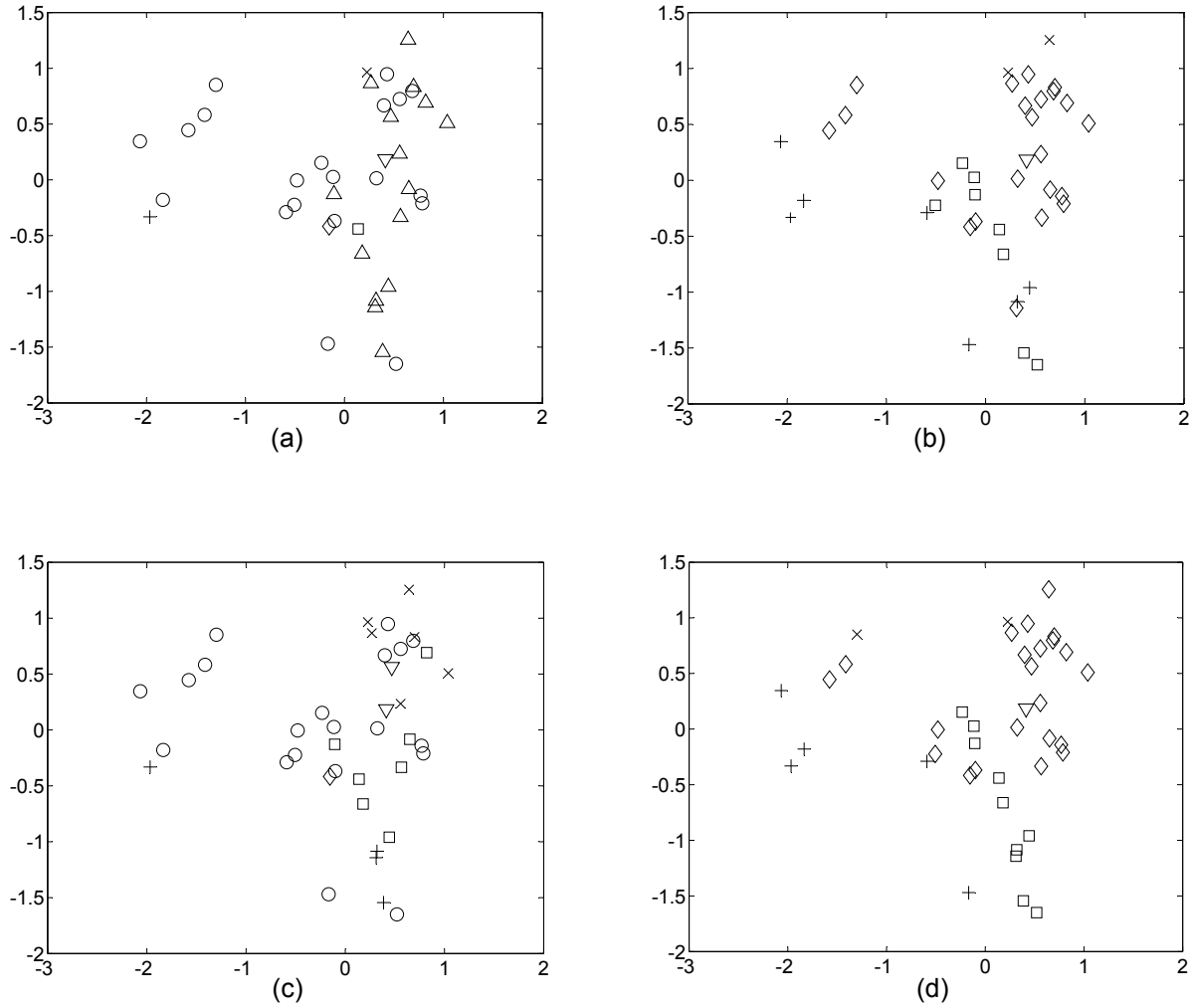


Figure 6-4: An example: comparison of SVM and LP algorithm on a small data set from ACE corpus. $\circ$ and $\triangle$ denote the unlabeled examples in the training set and the test set respectively, and other symbols ($\diamond, \times, \square, +$ and $\bigtriangledown$) represent the labeled examples with respective relation type sampled from training set.

An Example

In Figure 6-4, we selected 25 instances in the training set and 15 instances in the test set from the ACE corpus,which covered five relation types. Using the *Isomap* tool [2], the 40 instances with 229 feature dimensions are visualized in a two-dimensional space as the figure. We randomly sampled only one labeled example for each relation type from the 25 training examples as labeled data. Figure 6-4(a) and 6-4(b) show the initial state and ground truth result respectively. Figure 6-4(c) reports the classification result on test set by SVM ($accuracy = \frac{4}{15} = 26.7\%$), and Figure 6-4(d) gives the classification result on both the training set and test set by LP ($accuracy = \frac{11}{15} = 73.3\%$).

Comparing Figure 6-4(b) and Figure 6-4(c), we find that many examples are misclassified from class $\diamond$ to other class symbols. This may be caused that SVMs method ignores the intrinsic structure in data. For Figure 6-4(d), the labels of unlabeled examples are determined not only by nearby labeled examples, but also by nearby unlabeled examples, so using the LP strategy achieves a better performance than the local consistency based SVM strategy when the size of labeled data is quite small.

LP vs. Bootstrapping

In (Zhang, 2004), they perform relation classification on ACE corpus with bootstrapping on top of SVM. To compare with their proposed Bootstrapped SVM algorithm, we use the same feature stream setting and randomly selected 100 instances from the training data as the size of initial labeled data.

Table 6.3 lists the performance of the bootstrapped SVM method from (Zhang, 2004) and LP method with 100 seed labeled examples for relation type classification

[2]The tool is available at http://isomap.stanford.edu/.

Table 6.3: Comparison of the performance of the bootstrapped SVM method by Zhang (2004) and LP method with 100 seed labeled examples for relation type classification task.

	Bootstrapping			LP_{JS}		
Relation type	P	R	F	P	R	F
ROLE	78.5	69.7	73.8	81.0	74.7	77.7
PART	65.6	34.1	44.9	70.1	41.6	52.2
AT	61.0	84.8	70.9	74.2	79.1	76.6
SOC	47.0	57.4	51.7	45.0	59.1	51.0
NEAR	–	–	–	13.7	12.5	13.0

task. We can see that the LP algorithm outperforms the bootstrapped SVM algorithm on four of the relation type classification tasks, and performs comparably on the relation "SOC" classification task.

6.6 Discussion

In this Chapter,we have investigated a graph-based semi-supervised learning approach for relation extraction problem. Experimental results showed that the LP algorithm performs better than SVM and bootstrapping. We have some findings from these results:

The LP based relation extraction method can use the graph structure to smooth the labels of unlabeled examples. Therefore, the labels of unlabeled examples are determined not only by the nearby labeled examples, but also by nearby unlabeled examples. For supervised methods, e.g., SVM, very few labeled examples are not

Table 6.4: Comparison of the performance of previous methods on ACE RDC task.

	Method
Culotta et al.	Tree kernel based
Kambhatla	Feature based, Maximum Entropy model
Zhou et al.	Feature based, SVM model

	Relation Detection			Relation Detection and Classification					
				On Types			on Subtypes		
	P	R	F	P	R	F	P	R	F
Culotta et al.	81.2	51.8	63.2	67.1	35.0	45.8	-	-	-
Kambhatla	-	-	-	-	-	-	63.5	45.2	52.8
Zhou et al.	84.8	66.7	74.7	77.2	60.7	68.0	63.1	49.5	55.5

enough to reveal the structure of each class. Therefore they can not perform well, since the classification hyperplane was learned only from few labeled data and the coherent structure in unlabeled data was not explored when inferring class boundary. Hence, our LP-based semi-supervised method achieves a better performance on both relation detection and classification when only few labeled data are available.

Currently most of the works on the RDC task of ACE focused on supervised learning methods (Culotta and Soresen, 2004; Kambhatla, 2004; Zhou et al., 2005). Table 6.4 lists a comparison on the relation detection and classification of these methods. Zhou et al. (2005) reported the best result as 63.1%/49.5%/55.5 in *Precision/Recall/F-measure* on the relation subtype classification using feature based method, which outperforms tree kernel based method by Culotta and Soresen (2004). Compared with Zhou et al. (2005)'s method, the performance of our LP-based method is slightly

lower. It may be due to that we used a much simpler feature set. The current experiment focuses on the investigation of a graph based semi-supervised learning algorithm for relation extraction. We could use more effective feature sets (Zhou et al., 2005) or kernel based similarity measure with LP for relation extraction in future to do further comparison.

The proposed semi-supervised algorithm inevitably faces its limitations. Since the algorithm resolved the problem by exploiting the manifold structure (cluster structure) in data, the underlying definition of the manifold structure of the added unlabeled data will affect the result of label propagation. If the added unlabeled data holds a clarity manifold, it would help us to propagate labels through unlabeled examples more accurately, otherwise it will make it worse. Another limitation lies in the selection of initial seed examples. Just as other semi-supervised learning algorithms, it is very subjective to determine how to select these seeds and how many seeds are to be selected. Furthermore, our model has not yet handled a large amount of data and thus did not address the scalability issue. But, the graph-based algorithm needs to calculate the similarity among all examples, which accounts for much computational cost. Because semi-supervised learning is useful when the size of unlabeled data is large, this is clearly a problem.

6.7 Summary

This chapter approaches the problem of semi-supervised relation extraction using a label propagation algorithm. It represents labeled and unlabeled examples and their distances as the nodes and the weights of edges of a graph, and tries to obtain a labeling function to satisfy two constraints: 1) it should be fixed on the labeled

nodes, 2) it should be smooth on the whole graph. In the classification process, the labels of unlabeled examples are determined not only by nearby labeled examples, but also by nearby unlabeled examples. Our experimental results demonstrated that this graph based algorithm can achieve a better performance than SVM when only very few labeled examples are available, and also outperforms the bootstrapping method for relation extraction task.

In the next two chapters we will further investigate unsupervised solutions for extracting relations between entity mention pairs so that we can fully automatic the relation extraction task.

Chapter 7

An Unsupervised Model for Relation Extraction

The previous chapter presented a semi-supervised model for relation extraction. Although it reduces the requirement of a large amount of manually labeled data for supervised learning based approaches, one common feature of semi-supervised learning based method for relation extraction is that they still need to pre-define some initial seeds for any particular relation, and then to derive further relations from the seeds. However, to decide how to select these initial labeled data and how many labeled data are to be selected can be very subjective. To overcome the difficulties on the requirement of labeled data and enumeration of all class labels, unsupervised learning based methods have received more and more research interest.

In this chapter we will model relation extraction problem in an unsupervised learning manner. First, we give an overview of the main phases of the unsupervised learning based approach. Secondly, we introduce how to use the stability-based method to cluster the contexts. Thirdly, we report experimental results of the unsupervised

model with model order selection.

7.1 Model Unsupervised Relation Extraction Problem

Following the same assumption in the previous chapter of semi-supervised learning based model for relation extraction:

If two occurrences of entity mention pairs have the similarity context vectors, they tend to hold the same relation type.

we assume that pairs of entity mentions occurring in the similar context can be clustered and that each pair in a cluster is an instance of the same relation. Then unsupervised relation extraction problem can be formulated as a clustering task. Generally the basic idea of unsupervised relation extraction can be modeled as follows:

1. Tagging named entities in the text corpus;
2. Getting co-occurrence pairs of named entities and their context;
3. Measuring context similarities among pairs of named entities;
4. Making clusters of pairs of named entities;
5. Labeling the relation for each cluster of pairs of named entities.

7.1.1 Named entity tagging

Since our goal here is to achieve fully unsupervised learning, we do not need richly annotated corpora or any initial manually selected seeds. For the plain corpus, we only need a named entity (NE) tagger. As before, we use the ACE corpus, which already tags named entities in the corpus.

7.1.2 Context Collecting

We assume that for any two particular entity mentions $e_1 \in E_1$, and $e_2 \in E_2$, they may hold more than one kind of relation. So, we collect the contexts from a corpus in which e_1 and e_2 co-occur within a context window of d words in a sentence. Here,the context includes the words between, before and after them (in this chapter, following Hasegawa et al. (2004)'s work, we use only *words* as the features of context vectors). In fact, the approach also applies to the cases that e_1 and e_2 hold only one kind of relations, in such cases, we need to collect and accumulate the contexts as (Hasegawa et al., 2004).

7.1.3 Context Similarity among Entity Pairs

Following Hasegawa et al. (2004), we only compare entity pairs which have the same named entity types, e.g., one *PERSON-GPE* pair and another *PERSON-GPE* pair. We define a domain as a pair of named entity types, e.g., the *PERSON-GPE* domain. For example, we have to detect relations between *PERSON* and *GPE* in the *PERSON-GPE* domain.

In this study, cosine similarity is adopted to calculate the similarities between the context vectors of entity mention pairs. A context vector for each entity mention pairs

consists of the bag of words formed from all intervening words from all co-occurrences of two entity mention pairs. The cosine similarity $cosine(\theta)$ between context vector α and β is calculated by the following formula:

$$cosine(\theta) = \frac{\alpha \cdot \beta}{|\alpha||\beta|} \tag{7.1}$$

7.1.4 Context Clustering

In this phase, we cluster each context by the type of relation it represents. For a cluster c with a relation r, the entity mentions e_1 and e_2 whose context vector belongs to c can be regarded as holding the relation r. The most prevalent problem for context clustering is that we do not exactly know the number of relation types in advance. Moreover, since we do not have any labeled data at hand, we also can not learn this information from the labeled data. The previous methods (Hasegawa et al., 2004) for unsupervised relation extraction task also did not address this problem.

Compared with supervised and semi-supervised methods, Hasegawa et al. (2004)'s unsupervised approach for relation extraction can overcome the difficulties on requirement of a large amount of labeled data and enumeration of all class labels. Hasegawa et al. (2004)'s method is to use a hierarchical clustering method to cluster pairs of named entities according to the similarity of context words intervening between the named entities. However, the drawback of hierarchical clustering is that it required providing the cluster number by the user.

For a fully unsupervised model, we should achieve the order identification capability, so that we can exactly know the most likely number of relation types held in all entity mention pairs. In the next section and the next chapter, the discussion

describes the strategies we used to overcome this challenge.

7.1.5 Relation Labeling

After context clustering, each cluster can be regarded as a set of entity mention pairs which hold the same relation type. Hence, we need to select some representative words as the label of each relation cluster. In (Hasegawa et al., 2004), they simply select the frequent common words in a cluster to become the label of the relation. In our work, we use DCM (Discriminative Category Matching) scheme to identify discriminative label, which is also used in document classification (Fung et al., 2002) and weights the importance of a feature based on their distribution. For relation labeling, we do not address more in this thesis. The details can reference our previous work (Chen et al., 2005a).

In the next section we will introduce how to resolve the context clustering problem in our unsupervised model.

7.2 An Unsupervised Model with Order Identification Capability

Since we do not know how many relation types in advance and do not have any labeled relation training examples at hand, the problem of model order selection arises, i.e. estimating the "correct" number of clusters. In this chapter, the model selection capability is achieved by resampling based stability analysis, which has been successfully applied to several unsupervised learning problems (e.g. (Levine and Domany, 2001), (Lange et al., 2002), (Roth and Lange, 2003) , (Niu et al., 2004)).

Table 7.1: Model selection algorithm for relation extraction

Model Selection Algorithm for Relation Extraction:
Input: Corpus D tagged with Entities(E_1, E_2); Output: Model Order (number of relation types);
1. Collect the contexts of all entity pairs in the document corpus D, namely P;
2. Set the range (K_l, K_h) for the possible number of relation clusters;
3. Set estimated model order $k = K_l$;
4. Cluster all entity pairs set P into k clusters using stability analysis method;
5. Record k and the score of the merit of k, namely M_k;
6. If $k < K_h$, $k = k + 1$, go to step 4; otherwise, go to Step 7;
7. Select k which maximizes the score of the merit M_k;

To estimate the number of the clusters, we need a criterion to evaluate the merit for each possible number of clusters, and select the model order which maximizes the criterion. Formally, let k be the model order, we need to find k as follows:

$$k = \arg\max_{k}\{criterion(k)\} \tag{7.2}$$

Here, the criterion is set up based on resampling-based stability analysis. Table 7.1 shows the procedure of the model selection algorithm for unsupervised relation extraction.

The basic idea of stability based model selection is:

Solutions on two data sets from the same source should be similar!

This idea ensures that the clustering solution reflects structural properties of the data source, and that it will not be influenced too much by noise in the data. The general procedure for this idea is:

1. Draw two data sets X, X' from the same source;

2. Cluster both data sets using a clustering algorithm;

3. Compute agreement between both solutions.

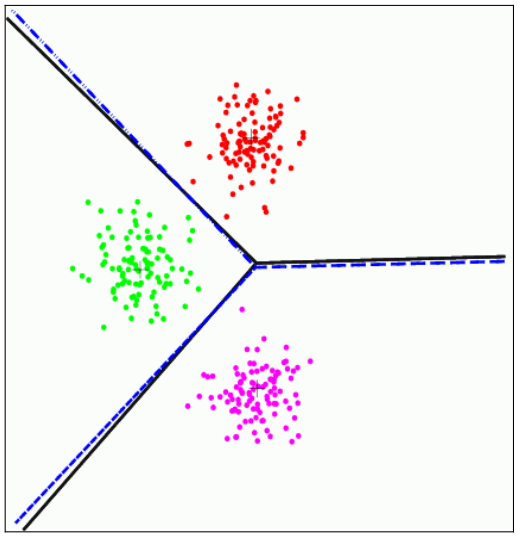
Then Stability := expected agreement of solutions.

As an example which shows how the wrong number of clusters can lead to instable solutions, consider the case in Figure 7-1: The data set is a mixture of Gaussians with 3 modes, and it is clustered with K-means. The picture shows the cluster boundaries for the case when $k = 3$ and when $k = 4$. As one can see, for $k = 4$, the two solutions are very different, because the mode which was split was a different one. We conclude that the agreement between two solutions for data sets from the same source is indicative for the number of clusters to be inferred. Moreover, having a stable solution is also highly preferable from a practical point of view.

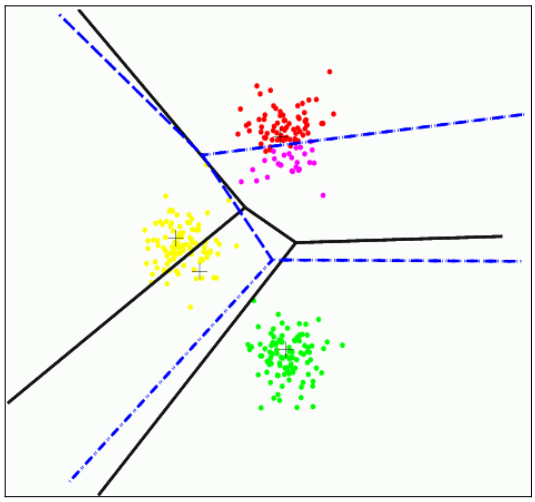
From this example,we also can conclude, if the model order is (not) appropriate for the data, then groupings on different data from the same source data are similar (dissimilar) with high probability.

However, the practical problem is that actually X' is not available. Thus, *resampling* acts as an alternative.

Let P^{μ} be a subset sampled from full entity pairs set P with size $\alpha|P|$ (α set as 0.9 in this study.), $C(C^{\mu})$ be $|P| \times |P|(|P^{\mu}| \times |P^{\mu}|)$ connectivity matrix based on



Stable Solution: K = 3



Unstable Solution: K = 4

Figure 7-1: An example for stability based model selection.

the clustering results on $P(P^\mu)$. Each entry $c_{ij}(c^\mu_{ij})$ of $C(C^\mu)$ is calculated in the following: if the entity pair $p_i \in P(P^\mu)$, $p_j \in P(P^\mu)$ belong to the same cluster, then $c_{ij}(c^\mu_{ij})$ equals 1, else 0. Then the stability is defined in Equation 7.3:

$$M(C^\mu, C) = \frac{\sum_{i,j} \delta^\mu_{i,j}}{\sum_{i,j} \delta_{i,j}} \tag{7.3}$$

where

$$\delta^\mu_{i,j} = \begin{cases} 1, & \text{if } C^\mu_{i,j} = C_{i,j} = 1, p_i \in P^\mu, p_j \in P^\mu; \\ 0, & \text{otherwise.} \end{cases} \tag{7.4}$$

$$\delta_{i,j} = \begin{cases} 1, & \text{if } C_{i,j} = 1, p_i \in P^\mu, p_j \in P^\mu; \\ 0, & \text{otherwise.} \end{cases} \tag{7.5}$$

Intuitively, $M(C^\mu, C)$ denotes the consistency between the clustering results on C^μ and C. The assumption is that if the cluster number k is actually the "natural" number of relation types, then clustering results on subsets P^μ generated by sampling should be similar to the clustering result on full entity pair set P. Obviously, the above function satisfies $0 \leq M \leq 1$.

It is noticed that $M(C^\mu, C)$ tends to decrease when increasing the value of k. Therefore for avoiding the bias that small value of k is to be selected as cluster number, we use the cluster validity of a random predictor ρ_k to normalize $M(C^\mu, C)$. The random predictor ρ_k achieved the stability value by assigning uniformly drawn labels to objects, that is, splitting the data into k clusters randomly. Furthermore, for each k, we tried q times. So, the normalized object function can be defined as Equation 7.6:

Table 7.2: Some context examples in two clusters of the output in the domain PER-ORG.

Cluster 1:
[PER] vice president of the [ORG]
[PER] president and chief operating officer of [ORG]
[PER] senior vice president of [ORG]
⋮
Cluster 2:
[PER] joined the communist -backed [ORG]
[PER] and joined a laborer's [ORG]
[PER] a partner in Blackstone, will join Host Marriott's [ORG]
⋮

$$M_k^{norm} = \frac{1}{q}\sum_{i=1}^{q} M(C_k^{\mu_i}, C_k) - \frac{1}{q}\sum_{i=1}^{q} M(C_{\rho_k}^{\mu_i}, C_{\rho_k}) \tag{7.6}$$

Normalizing $M(C^{\mu}, C)$ by the stability of the random predictor can yield values independent of k. The effect of such normalization can be observed from the experimental results (See Table 7.5). Table 7.3 shows the evaluation procedure of model order selection.

After the number of optimal clusters has been chosen, we adopted K-means algorithm for the clustering phase. The output of context clustering is a set of context clusters, each of them is supposed to denote one relation type. As an example, Table 7.2 lists two clusters with some context examples.

Table 7.3: Unsupervised algorithm for evaluation of model order selection

---

Function: criterion(k, P, q)

Input: cluster number k, entity pairs set P, and sampling frequency q;

Output: the score of the merit of k;

1. With k as input, perform K-means clustering analysis on pairs set P;

2. Construct connectivity matrix C_k based on above clustering solution on P;

3. Use random predictor ρ_k to assign uniformly drawn labels to each object in P;

4. Construct connectivity matrix C_{ρ_k} based on above clustering solution on P;

5. Construct q subsets of the full pairs set, by randomly selecting αN of the N original pairs, $0 \leq \alpha \leq 1$;

6. For each subset, perform the clustering analysis in Step $2, 3, 4$, and result C_k^{μ}, $C_{\rho_k}^{\mu}$;

7. Compute M_k^{norm} to evaluate the merit of k using Equation 7.6;

8. Return M_k^{norm};

Table 7.4: Three domains of entity pairs: frequency distribution for different relation subtypes

PER-ORG	num:786	ORG-GPE	num:262	ORG-ORG	num:580
Subtypes	Percentage	Subtypes	Percentage	Subtypes	Percentage
Management	36.39%	Based-In	46.56%	Member	27.76%
General-staff	29.90%	Located	35.11%	Subsidiary	19.83%
Member	19.34%	Member	11.07%	Part-Of	18.79%
Owner	4.45%	Affiliate-Partner	3.44%	Affiliate-Partner	17.93%
Located	3.28%	Part-Of	2.29%	Owner	8.79%
Client	1.91%	Owner	1.53%	Client	2.59%
Other	1.91%			Management	2.59%
Affiliate-Partner	1.53%			Other	1.21%
Founder	0.76%			Other	0.52%

7.3 Experimental Evaluations

In this experiment, we evaluate our proposed unsupervised model by comparing with the previous unsupervised approach (Hasegawa et al., 2004).

7.3.1 Experiment setup

Following Hasegawa et al. (2004)'s work , we constructed three subsets from ACE corpus for domains PER-ORG (person-organization), ORG-GPE (organization-gpe) and ORG-ORG (organization-organization) respectively. The details of these subsets

are given in Table 7.4, which are broken down by different relation types.

To verify our proposed method, we only extracted those pairs of entity mentions which have been tagged relation types in the given corpus for evaluation. Then the relation type tags were removed to test the unsupervised relation disambiguation. During the evaluation procedure, the relation type tags were used as ground truth classes.

Following Hasegawa et al. (2004)'s work, in this chapter we only use the *word* features to construct the context vectors. And the data preprocessing involves lowering the upper case characters, ignoring all words that contain digits or non alpha-numeric characters, removing words from a stop word list, stemming and filtering out low frequency words which appeared only once in the entire set.

7.3.2 Evaluation method for clustering result

When assessing the agreement between clustering result and hand-tagged relation types (ground truth classes), we encounter the problem that there was no relation type tags for each cluster in our clustering results.

To resolve the problem, we adopted a permutation procedure to assign different relation type tags to only $min(|EC|,|TC|)$ clusters, where $|EC|$ is the estimated number of clusters, and $|TC|$ is the number of ground truth classes (relation types). This procedure aims to find an one-to-one mapping function Ω from the TC to EC which is based on the assumption that for any two clusters, they do not share the same class labels. Under this assumption, there are at most $|TC|$ clusters which are assigned relation type tags. If the number of the estimated clusters is less than the number of the ground truth clusters, empty clusters should be added so that

$|EC| = |TC|$ and the one-to-one mapping can be performed.

With the estimated clusters and the ground truth classes, we construct a contingency table T, where each entry $t_{i,j}$ gives the number of the instances that belong to both the i-th cluster and j-th ground truth class. The mapping procedure can be formulated as the function:

$$\hat{\Omega} = \arg\max_{\Omega} \sum_{j=1}^{|TC|} t_{\Omega(j),j} \tag{7.7}$$

where $\Omega(j)$ is the index of the estimated cluster associated with the j-th class.

Given the result of one-to-one mapping, we adopt *F-measure* to evaluate the clustering result.

7.3.3 Experiments and Results

For comparison of the effect of the outer context of entity pairs, we set the middle context as everything between two entity mentions within a sentence and conducted three different settings of outer context window size for each domain. For example, the setting of "2" means that the intervening words between an entity mention pair together with the *two* words before the first entity mention and *two* words following the second entity mention constitute the context vector of the entity mention pair.

Table 7.5 shows the results of model order identification with unnormalized and normalized objective functions. The results show that the model order identification algorithm with the unnormalized function M_k^{unnorm} fail to identify the real number of relation types since the score of M_k^{unnorm} decreased when increasing the cluster number k and finally resulted in 2 clusters over all domains. On the other hand, the algorithm with the normalized function M_k^{norm} achieves the reasonable cluster

Table 7.5: Automatically determined the number of relation subtypes using different evaluation functions: M_k^{unnorm} is unnormalized objective function and M_k^{norm} is normalized objective function.

	PER-ORG			ORG-GPE			ORG-ORG		
Context Win Size	Real #	M_k^{unnorm}	M_k^{norm}	Real #	M_k^{unnorm}	M_k^{norm}	Real #	M_k^{unnorm}	M_k^{norm}
0	9	2	7	6	2	6	9	2	7
2	9	2	8	6	2	6	9	2	8
5	9	2	5	6	2	2	9	2	6

number around the real value over three domains.

From Table 7.5, we also can find that with the context window size setting, "2", the estimated number of the clusters equals or very close to the real number of classes, which is better than the setting without outer context. Furthermore, with context window size setting as "5", the estimated cluster number is the most far away from the real number of classes. It demonstrates that the close contextual words beyond (before and after) the entities may be appropriate features, which can help reflect the structure behind the contexts. However, when extending the outer context window size too much, it would tend to include more noisy features to disturb the relation disambiguation instead, as can be seen that the performance deteriorates.

Table 7.6 shows the performance of the clustering algorithm over three domains with different context window size settings. In this table, we compared the clustering results of our unsupervised model with the Hasegawa et al. (2004)'s context clustering algorithm. For the Hasegawa et al. (2004)'s clustering algorithm, i.e. hierarchical clustering, we specify the cluster number as the number of ground truth classes.

Table 7.6: Performance of the context clustering algorithm with various context window size settings over three domains.

	PER-ORG		ORG-GPE		ORG-ORG	
Context Window size	F1 (our method)	F2 (Hasegawa's method)	F1 (our method)	F2 (Hasegawa's method)	F1 (our method)	F2 (Hasegawa's method)
0	35.7	33.5	47.4	43.1	**41.0**	29.9
2	**39.4**	36.2	**50.7**	42.9	38.9	28.1
5	31.3	28.4	46.5	42.3	33.2	26.3

Comparing the *F-measure* result of two clustering methods, we can find that our method can achieve better or comparable performance obviously. The result confirms that the estimated model order of our method can reflect a preferable cluster structure corresponding to the distribution of various relation subtypes. In addition, in three domains, we can see that the best performance is achieved in the context window size setting, "0" or "2". On the other hand, the performance becomes worse when extending the context window too much, that is, when the context window size setting is "5". The reason is that extending the context too much may include more features, but at the same time, the noise also increases.

7.4 Discussion

In this chapter, we try to resolve the relation extraction task in an fully unsupervised manner. Compared with the existing unsupervised method (Hasegawa et al., 2004), there are several advantages in our approach.

Relation Types In (Hasegawa et al., 2004), each entity pair is treated as having

one and only one relation type, so they accumulated contexts of all occurrences of an entity pair. That is, only one context vector was generated for an entity pair. However, our proposed method is based on a more reasonable assumption that there may exist several relation types among different occurrences of an entity pair, so, we collect all instances of the occurrences of an entity pair, and represent each instance using a context vector. Then our task turns into disambiguating the relation types among the context occurrences of all entity pairs.

Context Clustering (Hasegawa et al., 2004) adopted a hierarchical clustering method to cluster the contexts. It is very difficult to determine the threshold for the similarity between clusters, like the appropriate number of clusters. In contrast, through model order selection we can estimate the "natural" number of relation types so that we do not need to manually pre-define any parameters during the clustering process.

Evaluation method In (Hasegawa et al., 2004), each cluster is mapped to one ground truth class simply by choosing the one which has the most overlap with it. But two clusters may be mapped to the same relation class, to avoid this bias, we try to find a one to one mapping from the estimated cluster to the ground truth classes.

However, the proposed unsupervised model achieved the above advantages at the expense of more computing time. This is because for each possible cluster number, the algorithm needs to perform resampling analysis iteratively, and to calculate the similarity among examples. These operations would increase time complexity of the algorithm.

7.5 Summary

In this chapter, we proposed an unsupervised model for relation disambiguation, using resampling based stability analysis. The advantages of the proposed approach includes that it does not need any manual labeling of the relation instances, it does not need to pre-define the number of the context clusters, or pre-specify the similarity threshold for the clusters.

Following the domain and feature setting in Hasegawa et al. (2004)'s work, we provide an experimental comparison between our method with Hasegawa et al. (2004)'s clustering algorithm and verify that our method can achieve a better performance and identify the "natural" number of relation types.

In the next chapter, we will propose another unsupervised model to further improve the performance of relation disambiguation and can be performed efficiently.

Chapter 8

An Improved Model for Unsupervised Relation Disambiguation

The previous chapter presented a unsupervised model for automatic relation extraction, which adopt resampling based stability analysis to achieve order identification capability. However, like the previous unsupervised method (Hasegawa et al., 2004), the model performed clustering in the original high dimensional space, which may take too much time to perform iterative operations of resampling and stability analysis, and also hard to identify non-convex clusters. Inevitably, we would like to find an improved model for unsupervised relation disambiguation to challenge these underling problems. In chapter 6, we have mentioned that graph-based algorithm is suitable for relation extraction task. Under the same model assumption that if it is true that two points with similar features tend to be in the same class, then graph-based methods can be used for unsupervised model for relation extraction. Since unlike

semi-supervised relation extraction, which has some initial labeled data to propagate the label information to those unlabeled vertices, then how to apply the graph strategy to resolve the unsupervised relation disambiguation problem? And how to obtain data representation in the low-dimensional space that can be easily clustered?

In this chapter we present another improved model for unsupervised relation disambiguation using graph-based methods. First, we formulate the unsupervised relation disambiguation problem using a graph-based strategy. Then we come to the second stage of this method: cluster these context vectors automatically. We will present how to apply the spectral clustering technique to resolve the task, which involves how to transform the clustering space and how to modify the K-means algorithm as the elongated K-means algorithm to adapt the problem. Finally we report experimental results comparing with other unsupervised methods for relation disambiguation.

8.1 Modeling Graph-based Unsupervised Relation Disambiguation Problem

Assume that two occurrences of entity pairs with similar contexts hold the same relation type. Thus unsupervised relation disambiguation problem can be formulated as partitioning collections of entity pairs into clusters according to the similarity of contexts, with each cluster containing only entity pair labeled by the same relation type.

Let $X = \{x_i\}_{i=1}^{n}$ be the set of context vectors of occurrences of all entity mention pairs, where x_i represents the context vector of the i-th occurrence, and n is the total number of occurrences of all entity mention pairs.

As described in the previous chapters, each occurrence of an entity mention pair can be denoted as follows:

$$R \rightarrow (C_{pre}, e_1, C_{mid}, e_2, C_{post}) \tag{8.1}$$

where e_1 and e_2 represent the entity mentions, and C_{pre},C_{mid},and C_{post} are the contexts before, between and after the entity pair respectively.

In this chapter, we also extract features from e_1, e_2, C_{pre}, C_{mid}, C_{post} to construct context vectors. And the feature set is the same as the feature set that we described in Chapter 5, which includes words, entity type, POS features, and chunking features. We combine the above lexical and syntactic features with their position information in the context to form the context vector. Before that, we filter out low frequency features which appeared only once in the entire set.

We represent each context vector of an entity pair as a vertex in an undirected graph. Each edge (i,j) in the graph is assigned a weight W_{ij} that reflects the similarity between two context vectors i and j. That is, the set of context vectors $X = \{x_i\}_{i=1}^{n}$ can be represented as a weighted graph $G(V, E)$, where $V = \{x_i\}$ and $E = \{W_{ij}\}$.

Hence, the relation disambiguation task for entity mention pairs can be defined as a partition of the graph so that entity mention pairs that are more similar to each other, e.g. labeled by the same relation type, belong to the same cluster. However, as we know, the graph partition is an NP-hard problem.

As a relaxation of such NP-hard discrete graph partitioning problem, spectral clustering is a possible solution, which represents the above similarity graph as a matrix. From the knowledge of linear algebra, the eigenvalues and eigenvectors of a matrix provide global information about its structure. The top eigenvectors of the

graph Laplacian can unfold the data manifold to form meaningful clusters. This is the intuition behind spectral clustering. Hence, spectral clustering technique computes eigenvalues and eigenvectors of a Laplacian matrix related to the given graph, and construct data clusters based on such spectral information. At the heart of this approach is a transformation of the original input into a set of orthogonal eigenvectors. Then it works in the space defined by the first few eigenvectors, using standard clustering techniques in the reduced space.

$$\begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \lambda \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \tag{8.2}$$

8.2 Context Clustering Using Spectral Clustering

In recent years, spectral clustering technique has received more and more attention as a powerful approach to a range of clustering problems. Among the efforts on spectral clustering techniques (Weiss, 1999; Kannan et al., 2000; Shi and Malik, 2000; Ng et al., 2001; Zha et al., 2001; Sanguinetti et al., 2005), we adopt a modified version of the algorithm by (Ng et al., 2001), which can provide us model order selection capability.

A significant challenge in any clustering task is to determine how many clusters should be created for the given data. While discriminating relation types, we face a similar question: how many relation types do entity pairs actually have? As we do not have any labeled relation training examples at hand, the problem of model order selection arises, i.e. estimating the "optimal" number of clusters. Formally, let k be

the model order, we need to find k as follows:

$$k = \arg\max_{k}\{criterion(k)\} \tag{8.3}$$

Here, the criterion is defined on the result of spectral clustering. The following discussion describes the various strategies we used to overcome this challenge in our experiments.

Table 8.1 shows the details of the whole algorithm for context clustering, which contains two main stages:

1. Transformation of clustering space (Step 1-4);
2. Clustering in the transformed space using the elongated K-means algorithm (Step 5-6).

8.2.1 Transformation of Clustering Space

In this step, we want to obtain data representation in a low-dimensional space that can be easily clustered.

The starting point of context clustering is to construct an *affinity matrix* A from the data, which is an $n \times n$ matrix encoding the distances between the various points. The nature of the affinity matrix is that "closer" vertices will get larger weights, as Figure 8-1 presents. The affinity matrix is then normalized to form a matrix L^1 by conjugating with the diagonal matrix $D^{-1/2}$ which has as entries the square roots of the sum of the rows of A. This is to take into account the different spread of the

[1]There are several variations in the definition of L: some authors prefer to use $I - L$, some others set to zero the diagonal entries in A. These differences do not significantly alter the algorithm.

Table 8.1: Context clustering using spectral-based clustering technique.

Algorithm: Context Clustering with Model Order Selection

Input: A set of context vectors $X = \{x_1, x_2, ..., x_n\}$, $X \in \Re^{n \times d}$;
Output: Clustered data and number of clusters;

1. Construct an affinity matrix $A \in R^{n \times n}$ by

$$A_{ij} = \begin{cases} \exp(-\frac{s_{ij}^2}{\sigma^2}), & \text{if } i \neq j; \\ 0, & \text{if } i = j. \end{cases} \tag{8.4}$$

   Here, s_{ij} is the similarity between x_i and x_j calculated by Cosine similarity measure. And the free distance parameter σ is used to scale the weights.

2. Normalize the affinity matrix A to create the matrix

$$L = D^{-1/2} A D^{-1/2} \tag{8.5}$$

   where D is a diagonal matrix whose (i,i) element is the sum of A's ith row, i.e.,

$$D = diag(\sum_{j=1}^{d} A_{ij}) \tag{8.6}$$

3. Set $q = 2$;

4. Compute q eigenvectors of L with the greatest eigenvalues. Arrange them in a matrix Y.

5. Perform elongated K-means with $q+1$ centers on Y, initializing the $(q+1)$-th mean at the origin;

6. If the $q+1$-th cluster contains any data points, then there must be at least an extra cluster; set $q = q+1$ and go back to step 4. Otherwise, algorithm stops and outputs clustered data and number of clusters.

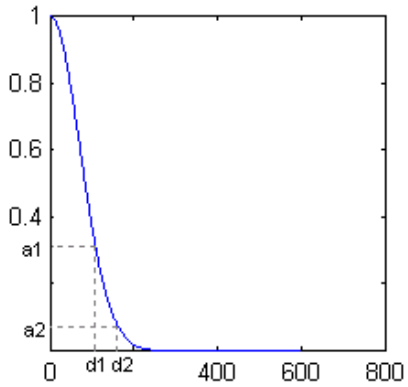
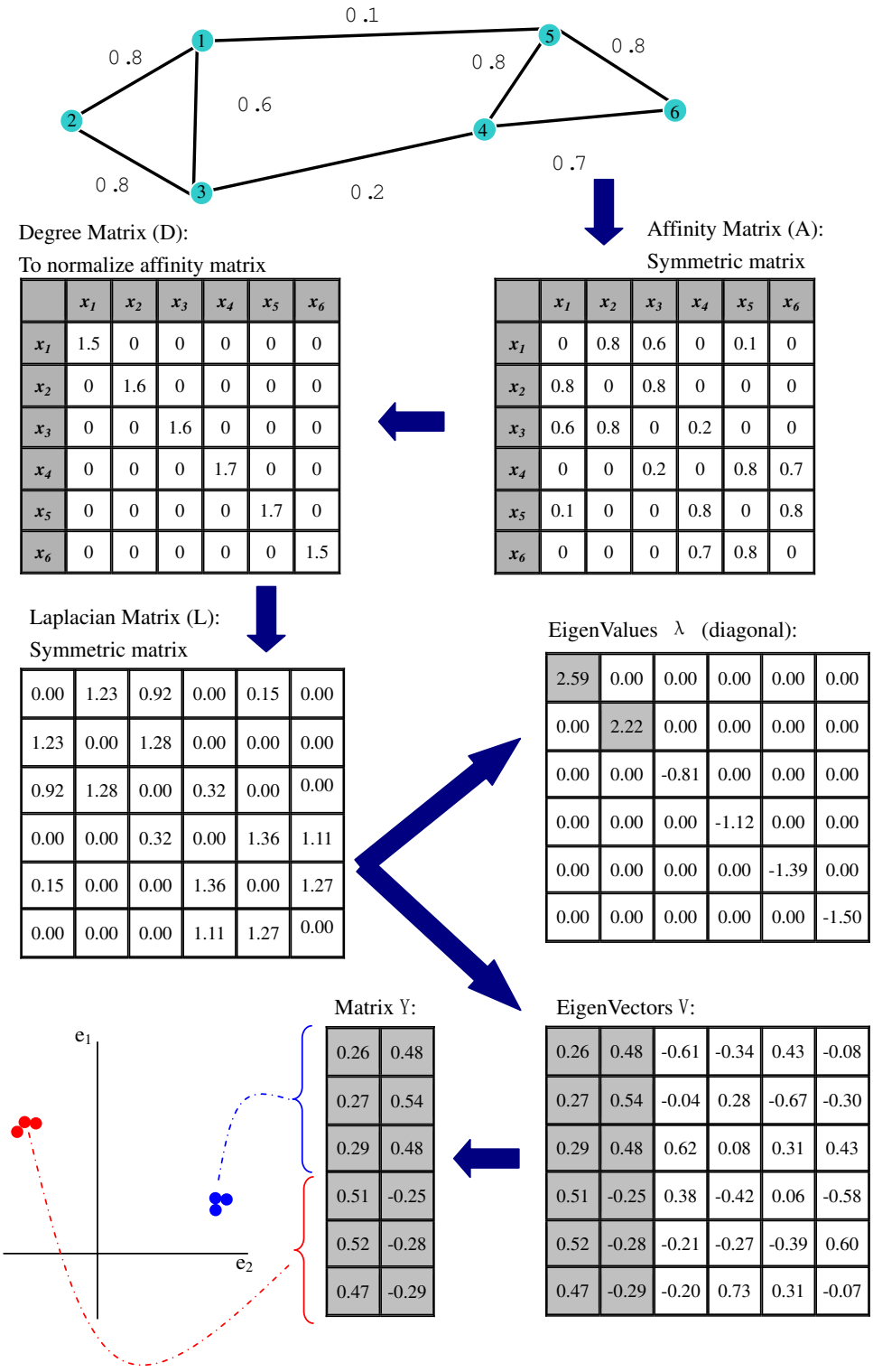


Figure 8-1: Nature of the affinity matrix

various clusters (points belonging to more rarified clusters will have lower sums of the corresponding row of A). Figure 8-2 shows an example about the above matrix representation for spectral clustering algorithm. It is straightforward to deduce that the matrix L is a symmetric matrix. For symmetric matrices, they have an important property, i.e., eigenvectors for distinct eigenvalues are orthogonal.

Let K be the true number of clusters present in the dataset. We have discussed that the top eigenvectors of the graph's Laplacian can unfold the data manifold to form meaningful clusters. Hence, if K is known beforehand, the first K eigenvectors of L (the ones corresponding to the largest eigenvalues) will be computed and arranged as columns in a $n \times K$ matrix Y. Each row of Y corresponds to a context vector of the entity pair, and the above process can be considered as transforming the original context vectors in a d-dimensional space to new context vectors in the K-dimensional space. Therefore, the rows of Y will cluster upon mutually orthogonal points on the K dimensional sphere, rather than on the coordinate axes (that is, it has reduced dimension from $n \times n$ to $n \times K$).



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	0	0	0	0	0
x_2	0	1.6	0	0	0	0
x_3	0	0	1.6	0	0	0
x_4	0	0	0	1.7	0	0
x_5	0	0	0	0	1.7	0
x_6	0	0	0	0	0	1.5

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

0.00	1.23	0.92	0.00	0.15	0.00
1.23	0.00	1.28	0.00	0.00	0.00
0.92	1.28	0.00	0.32	0.00	0.00
0.00	0.00	0.32	0.00	1.36	1.11
0.15	0.00	0.00	1.36	0.00	1.27
0.00	0.00	0.00	1.11	1.27	0.00

2.59	0.00	0.00	0.00	0.00	0.00
0.00	2.22	0.00	0.00	0.00	0.00
0.00	0.00	-0.81	0.00	0.00	0.00
0.00	0.00	0.00	-1.12	0.00	0.00
0.00	0.00	0.00	0.00	-1.39	0.00
0.00	0.00	0.00	0.00	0.00	-1.50

0.26	0.48
0.27	0.54
0.29	0.48
0.51	-0.25
0.52	-0.28
0.47	-0.29

0.26	0.48	-0.61	-0.34	0.43	-0.08
0.27	0.54	-0.04	0.28	-0.67	-0.30
0.29	0.48	0.62	0.08	0.31	0.43
0.51	-0.25	0.38	-0.42	0.06	-0.58
0.52	-0.28	-0.21	-0.27	-0.39	0.60
0.47	-0.29	-0.20	0.73	0.31	-0.07

Figure 8-2: An example of matrix representation for spectral clustering algorithm

8.2.2 The elongated K-means algorithm

Once we build the embedded space from the eigenvectors corresponding to the k largest eigenvalues, we can apply clustering algorithm on the matrix Y (treat each row as a context vector of the entity mention pair).

As the step 5 of Table 8.1 shows, we perform an elongated K-means algorithm to fulfill the objective. In this algorithm, the clustering result of elongated K-means algorithm is also used to detect whether the number of clusters selected q is less than the true number K, and allows one to iteratively obtain the number of clusters so that we can achieve order identification capability.

Why do we not just apply K-means algorithm directly?

We have mentioned that if we know the true number of clusters K present in the dataset, the rows of the matrix Y that has as columns the clustering eigenvectors will cluster upon mutually orthogonal points on the K dimensional sphere. Consider the case when the number of clusters q is less than the true cluster number K present in the dataset. In such a situation, taking the first $q < K$ eigenvectors, we will be selecting a q-dimensional subspace in the clustering space, whose position will in general bear no relation to the clusters. As the rows of the K eigenvectors clustered along mutually orthogonal vectors, their projections in a lower dimensional space will cluster along radial directions. Therefore, the general picture will be of q clusters elongated in the radial direction, with possibly some clusters very near the origin (when the subspace is orthogonal to some of the discarded eigenvectors).

Hence, given the elongated nature of the clusters, the K-means algorithm is modified as the elongated K-means algorithm to downweight distances along radial di-

rections and penalize distances along transverse directions. Explicitly, the elongated K-means algorithm computes the distance of point x from the center c_i as follows:

- If the center is not very near the origin, $c_i^T c_i > \epsilon$ (ϵ is a parameter to be fixed by the user), the distances are calculated as:

$$edist(x, c_i) = (x - c_i)^T M (x - c_i) \tag{8.7}$$

where

$$M = \frac{1}{\lambda}(I_q - \frac{c_i c_i^T}{c_i^T c_i}) + \lambda \frac{c_i c_i^T}{c_i^T c_i} \tag{8.8}$$

λ is the *sharpness* parameter that controls the elongation (the smaller, the more elongated the clusters) [2].

- If the center is very near the origin, $c_i^T c_i < \epsilon$, the distances are measured using the Euclidean distance.

In this way, if a center is within a cluster, all the points in the cluster will be very near to it, while points in another cluster (i.e. along another radial direction) will be judged to be further from that center than from the original.

In each iteration of procedure in Table 8.1, elongated K-means is initialized with q centers corresponding to data points in different clusters and one center in the origin. The algorithm then will drag the center in the origin towards one of the clusters accounted for. We compute the next eigenvector (thus increasing the dimension of the clustering space to $q + 1$) and repeat the procedure. Eventually, when one reach as many eigenvectors as the number of clusters present in the data, no points will be

[2] In this paper, the *sharpness* parameter λ is set to 0.2

assigned to the center at the origin, leaving the cluster empty. This is the signal to terminate the algorithm.

8.2.3 An example

Figure 8-3 visualizes the clustering result of three circle dataset using K-means and spectral-based clustering. Figure 8-3(a) is the visualization of three circle dataset.

From Figure 8-3(b), we can see that K-means does not separate the non-convex clusters in three circle dataset successfully since it is prone to local minimal.

For spectral-based clustering, as the algorithm described, initially (set $q = 2$), we took the two eigenvectors of L with largest eigenvalues, which gave us a two-dimensional clustering space. Then to ensure that the two centers are initialized in different clusters, one center c_1 is set as the point that is the farthest from the origin, while the other center c_2 is set as the point that simultaneously farthest the first center c_1 and the origin. As we know that there will be at least two clusters in the plane, this initialization guarantees that the second center is set at a point in a different cluster.

Then we add a third center c_3 at the origin. Because we are going to do an elongated K-means clustering, each center is considered closer to points that lie along the same radial line than to points that lie off this line. For this reason we find that the first two centers will not easily be moved away from the two clusters they started in. However, as c_3 is set in the origin, distances from it will be measured using the standard Euclidean distance, and this will mean that the points of the third cluster will be assigned to it (as their Euclidean distance from the origin is smaller than their elongated distance from another cluster). The consequence of this is that c_3 gets

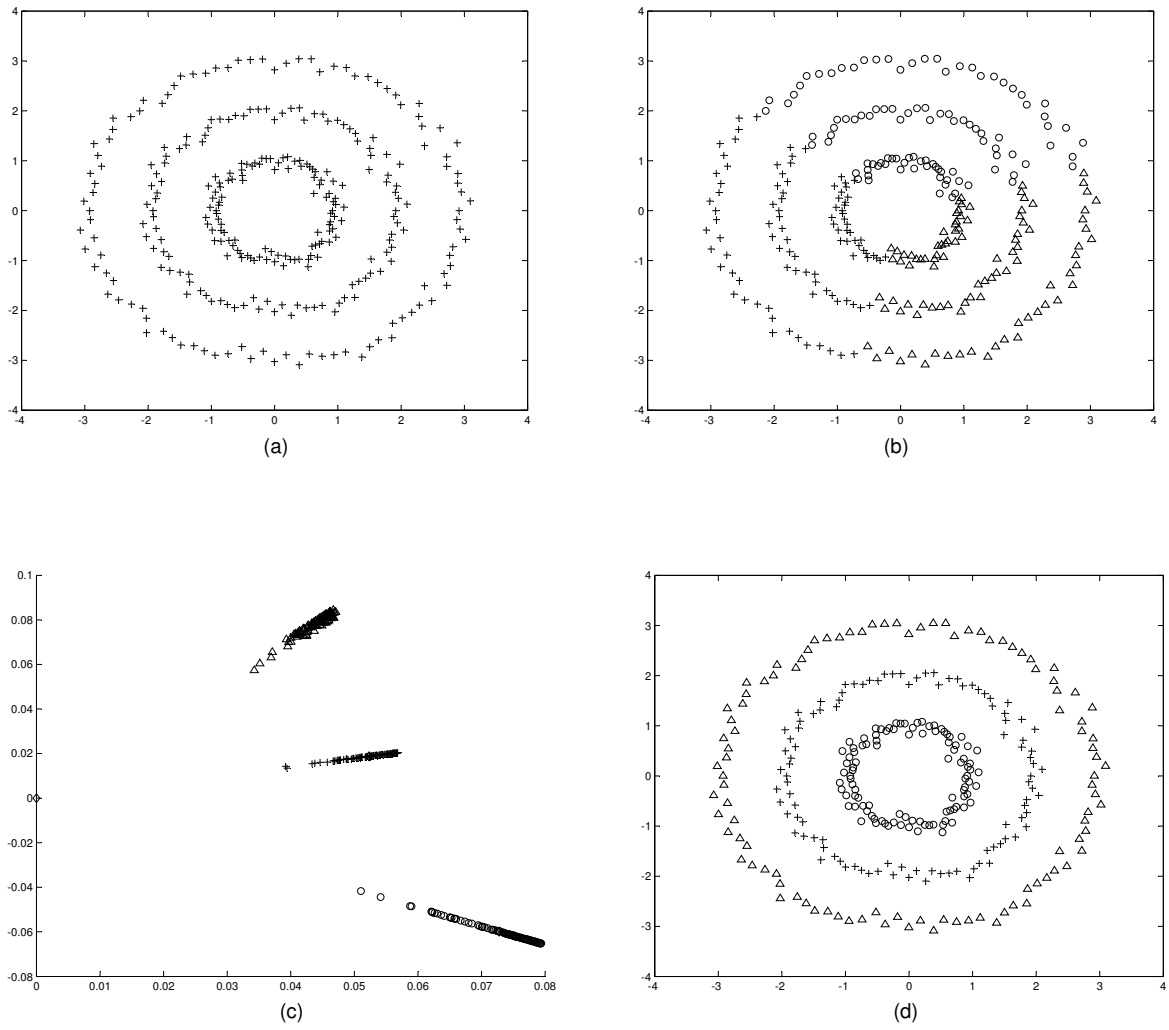


Figure 8-3: An example:(a) The three circle dataset. (b) The clustering result using K-means; (c) Three elongated clusters in the 2D clustering space using spectral clustering: two dominant eigenvectors; (d) The clustering result using spectral-based clustering (σ^2=0.05). ($\triangle$,$\circ$ and + denote examples in different clusters)

dragged towards the third cluster, and we achieve the clustering we desired, as each of the clusters in the 2D clustering space of Figure 8-3(c) corresponds to one of the concentric circles in Figure 8-3(a).

When iterating the algorithm further, all three clusters will have a mean vector initialized at one of their points. Therefore, the assignment rule of K-means will force the fourth center c_4 (initialized at the origin) to have no points assigned to it. This will be taken as the termination signal. The final result is visualized in Figure 8-3(d), which exploits manifold structure (cluster structure) in data.

8.3 Experiments and Results

8.3.1 Data Setting

Like the previous chapters, our proposed unsupervised relation extraction is evaluated on the ACE corpus. To verify our proposed method, we only collect those pairs of entity mentions in the devtest set of the given corpus. Then the relation type tags were removed to test the unsupervised relation disambiguation. During the evaluation procedure, the relation type tags were used as ground truth classes. Table 4.1 lists the types and subtypes of relations, along with their frequency of occurrence in the ACE training set and test set.

To verify the effectiveness of the improved model for unsupervised relation disambiguation, we use the same evaluation method for clustering as Chapter 7. Firstly, we find one-to-one mapping between the clustering result and hand-tagged relation types (ground truth classes), and then given the result of one-to-one mapping using Precision, Recall and F-measure to evaluate the clustering result.

Table 8.2: Contribution of different features

Features	cluster number	Precision	Recall	F-measure
Words	15	41.6	30.2	34.9
+Entity Type	18	40.3	42.5	41.5
+POS	18	37.8	46.9	41.8
+Chunking Infomation	19	43.5	49.4	46.3

8.3.2 Experimental Design

We perform our unsupervised relation extraction on the devtest set of ACE corpus and evaluate the algorithm on relation subtype level. Firstly, we observe the influence of Different Feature set and Context Window Size. Secondly, to verify the effectiveness of our method, we further compare it with other unsupervised methods.

Contribution of Different Features

As the previous section presented, we incorporate various lexical and syntactic features to extract relation. To measure the contribution of different features, we report the performance by gradually increasing the feature set, as Table 8.2 shows.

Table 8.2 shows that all of the four categories of features contribute to the improvement of performance more or less. Firstly, using word features only achieves the performance of 41.6%/30.2%/34.9 in *Precision/Recall/F-measure*. Secondly, the addition of entity type feature is very useful, which improves *F-measure* by 6.6. Thirdly, adding POS features can increase *F-measure* score but does not improve very much. Finally, chunking features also show their great usefulness with increasing *Precision/Recall/F-measure* by 5.7%/2.5%/4.5. With the addition of chunking

Table 8.3: Performance of context clustering with different context window size setting

Context Window Size	cluster number	Precision	Recall	F-measure
0	18	37.6	48.1	42.2
2	19	43.5	49.4	46.3
5	21	29.3	34.7	31.7

features, we also acquire the closer estimate cluster number to the truth number of ACE relation subtypes, 24, although it is not equal to this prior knowledge exactly.

Since these features are all helpful for the relation disambiguation task. We continue to combine all these features to do all other evaluations in our experiments.

Setting of Context Window Size

We have mentioned in Section 8.1 that the context vectors of entity pairs are derived from the contexts before, between and after the entity mention pairs. Hence, we have to specify the three context window size first. As before, we set the mid-context window as everything between the two entity mentions, and the context window size for pre- and post- context is defined as before.

For comparison of the effect of the outer context of the entity mention pair, we conducted three different settings of context window size (0, 2, 5) as Table 8.3 shows. From this table we can find that with the context window size setting, 2, the algorithm achieves the best performance of 43.5%/49.4%/46.3 in *Precision/Recall/F-measure*. With the context window size setting, 5, the algorithm achieves the closest estimated cluster number to the truth number of ground truth classes. However, it also results the worst performance. Hence, we can say that the setting of context window size

as “5” is not suitable because extending the context too much may include more features, but at the same time, the noise also increases, which may confuse the manifold structure. From this observation, we also can tell that characteristics of an relation instance is mainly determined by the most neighboring contexts around the two entity mentions.

Comparison with other Unsupervised methods

In this experiment, we explore the effectiveness of our unsupervised methods compared to other unsupervised methods. Table 8.4 is the performance using various context clustering techniques and feature sets.

In (Hasegawa et al., 2004), they preformed unsupervised relation extraction based on hierarchical clustering and they only used word features between entity mention pairs to construct context vectors. We reported the clustering results using the same clustering strategy as Hasegawa et al. (2004) proposed. In Table 8.4, Hasegawa's Method 1 means the test only used the word feature as Hasegawa et al. (2004)'s work, while Hasegawa's Method 2 means the test used the same feature set as our methods, which includes various lexical and syntactic features described in Chapter 5. In both tests, we specified the cluster number as the number of ground truth classes.

We also tried the relation disambiguation problem using the standard clustering technique, K-means algorithm, where we adopted the same feature set defined in our proposed method to cluster the context vectors of entity mention pairs and prespecified the cluster number as the number of ground truth classes.

In Table 8.4, we also report the result of our proposed unsupervised model in the previous chapter, which we will call “Our Proposed Method 1(Stability based)”. We

Table 8.4: Performance of various unsupervised methods for relation disambiguation.

	Precision	Recall	F-measure
Hasegawa's Method 1	38.7	29.8	33.7
Hasegawa's Method 2	37.9	36.0	36.9
K-means based Method	34.3	40.2	36.8
Our Proposed Method 1 (Stability based)	40.9	44.5	42.6
Our Proposed Method 2 (Spectral based)	43.5	49.4	46.3

call the unsupervised model proposed in this chapter as "Our Proposed Method 2 (Spectral-based)". Both of our proposed methods adopt the feature set we described in Chapter 5.

As the result shows, both of our proposed methods for context clustering clearly achieve better performance than Hasegawa et al. (2004)'s method and K-means based clustering method. Firstly, for Hasegawa et al. (2004)'s method, we find that the test using lexical and syntactic feature set outperforms the test using only word feature. This result again validates that the incorporation of various lexical and syntactic features is effective even for standard clustering method. Secondly, using the same feature set information, our proposed method1 which used resampling based stability analysis outperforms Hasegawa et al. (2004)'s method and another K-means clustering based methods by 5.7 and 5.8 in *F-measure* respectively. Finally, the best result for context clustering is achieved by our Proposed Method2 based on spectral clustering as 43.5%/49.4%/46.3 in *Precision/Recall/F-measure*, which is also better than our proposed method1 based on stability clustering. These findings support our assumption that the graph based method is effective for the relation extraction task.

8.3.3 Discussion

In this chapter, we have shown that the modified spectral clustering technique, with various lexical and syntactic features derived from the context of entity pairs, performed well on the unsupervised relation disambiguation problem. Our experiments show that we can estimate the cluster number without any labeled instances. We notice that the estimated cluster number is less than the number of ground truth classes in most cases. The reason for this phenomenon may be that some relation types can not be easily distinguished using the context information only. For example, the relation subtypes "Located", "Based-In" and "Residence" are difficult to disambiguate even for human experts to differentiate. Hence, the instances belong these subtypes may have similar context information and easily be recognized as the same clusters wrongly.

The results also show that various lexical and syntactic feature set contains useful information for the relation extraction task. Specifically, although we did not use the dependency tree and full parse tree information as other supervised methods (Miller et al., 2000; Culotta and Soresen, 2004; Kambhatla, 2004; Zhou et al., 2005), the incorporation of simple features, such as words and chunking information, can still provide complement information for capturing the characteristics of entity pairs. Another observation from the results is that extending the outer context window of the entity mention pair too much may not improve the performance since the process may incorporate more noise information and confuse the manifold structure.

As regards to the context clustering technique, our spectral-based clustering method performs better than other direct clustering methods, such as Hasegawa et al. (2004)'s Hierarchical clustering or K-means clustering. Since the spectral-based algorithm

works in a transformed space of low dimensionality, data can be easily clustered so that the algorithm can be implemented with better efficiency and speed. And the performance using spectral-based clustering can be improved due to the reason that spectral-based clustering overcomes the drawback of K-means clustering (prone to local minima) and may find non-convex clusters consistent with human intuition.

Currently most of works on the RDC task of ACE focused on supervised learning methods. Table 6.4 lists a comparison of these methods on relation detection and relation classification. (Zhou et al., 2005) reported the best result as 63.1%/49.5%/55.5 in *Precision/Recall/F-measure* on the extraction of ACE relation subtypes using feature based method, which outperforms tree kernel based method by (Culotta and Soresen, 2004). Although our unsupervised method still can not outperform these supervised methods, from the point of view of unsupervised resolution for relation extraction, our approach already achieves best performance of 43.5%/49.4%/46.3 in *Precision/Recall/F-measure* compared with other clustering methods.

8.4 Summary

In this chapter, we resolve the unsupervised relation disambiguation problem from the point of view of graph based method, by using spectral-based clustering technique with diverse lexical and syntactic features derived from context. It works by calculating eigenvectors of an adjacency graph's Laplacian to recover a submanifold of data from a high dimensional space, and then performing cluster number estimation on a transformed space defined by the first few eigenvectors. The advantage of our method is that it doesn't need any manually labeled relation instances, and pre-definition the number of the context clusters. This method may help us find non-convex clusters

and perform clustering effectively and efficiently. Experiment results on the ACE corpus show that our method achieves a better performance than other unsupervised methods. In the experiments we also examined the utility of the features in the unsupervised model and found out the different contribution of each feature.

Chapter 9

Conclusions and Future Work

The purpose of our thesis is to find effective semi-supervised and unsupervised learning models for the automatic relation extraction task. The traditional semi-supervised models are based on the local consistency assumption that examples close to labeled examples within the same class will have the same labels. As a result the affinity information among unlabeled examples can not be fully explored. Furthermore, the previous unsupervised models cannot determine the "natural" number of relation types among entity mention pairs and are unable to to handle non-convex clusters.

The thesis has confirmed our hypothesis that the need of a large amount of labeled data can be avoided for automatic relation extraction task. The main contribution of this thesis is that it presents graph based models for semi-supervised and unsupervised relation extraction task to overcome the above limitations of the previous works.

We will now summarize and highlight the significance of the research work that has been discussed in the previous chapters and will discuss some potential directions for future work.

9.1 Main Contributions

The thesis has the following contributions:

The construction of the graph based model for Semi-supervised relation extraction

With an aim to address the problems of the conventional models for relation extraction, this thesis proposes, for the first time to the best of our knowledge, graph based model to do relation extraction. Actually, the assumption of graph-based methods, that two points with similar features tend to be in the same class, fit the problem structure of relation extraction. As stated in Chapter 6, we proposed a Label Propagation (LP) based semi-supervised learning algorithm to learn from both labeled and unlabeled data. This algorithm works by representing labeled and unlabeled examples as vertices in a connected graph, then propagating the label information from any vertex to nearby vertices through weighted edges iteratively, finally inferring the labels of unlabeled examples after the propagation process converges.

The experimental results on the ACE corpus showed that our LP-based semi-supervised method achieves a better performance than SVM and another bootstrapping method based on SVM by Zhang (2004) on both relation detection and classification tasks when only few labeled data is available. The results also showed that our method achieves a comparable performance to SVM using the full set of the available ACE training examples. It is possible that, for supervised method (SVM) and bootstrapping method, too few labeled examples are not enough to reveal the structure because the classification hyperplane was learned only from few labeled data and the coherent structure in unlabeled data was not explored when inferring the class boundary. The findings indicated that our method can overcome the problem of not

having enough manually labeled relation instances for supervised relation extraction methods.

The achievement of order identification capability in unsupervised model

Chapter 7 modeled relation extraction problem in an unsupervised learning manner and gave an overview of the main phases of an unsupervised approach. Specifically, in this chapter, we introduced an unsupervised learning algorithm based on model order identification for automatic relation extraction. We have confirmed our hypothesis that model order identification can be achieved by resampling based stability analysis. The main idea behind the stability based model selection is that solutions on two data sets from the same source should be similar. Actually, previous works did not addressed model selection problem in unsupervised manner for relation extraction. Hence, this is a significant improvement over the unsupervised learning technique for relation extraction problem compared with the existing work by Hasegawa et al. (Hasegawa et al., 2004). Experiments results showed that we can infer the number of relation types between entity mention pairs automatically. With the estimated "natural" number of relation types, our method also outperforms the other unsupervised methods.

The improvement of unsupervised relation disambiguation using graph based model

Chapter 8 further investigated the unsupervised learning solution for relation extraction. Unlike Hasegawa et al. (2004)'s work, we also allow multiple relation to be captured for the same entity pair which leads to the need to do relation disambiguation. Enlightened by the graph based model for semi-supervised relation extraction in

Chapter 6, we modelled the unsupervised relation disambiguation problem as a graph partitioning problem. As a relaxation of such NP-hard discrete graph partitioning problem, we proposed a novel application of spectral clustering technique to detect and classify relation instances of entity pairs. Compared with the stability based method described in Chapter 7, the spectral-based algorithm can be implemented with much more efficiency and speed. It is due to space transformation from the original high dimensionality to a low dimensionality. Experimental results also showed that the spectral based method can improve the performance of context clustering.

Currently most of work on the RDC task of ACE focused on supervised learning methods. Although the experiments compared with these methods showed that our method still cannot outperform these supervised methods, from the point of view of unsupervised relation type disambiguation, our approach already achieves the best performance compared with other unsupervised based methods. The reason is that spectral clustering is likely to find non-convex clusters which traditional clustering algorithms cannot obtain. As a result, this efficient approach is a big step towards automatic relation extraction without any human intervention.

Knowledge representation for automatic relation extraction

Chapter 5 explores the knowledge representation issue in the our automatic relation extraction models. Our thesis proposes to represent each relation instance using the context information before, between and after an entity mention pair and the two entity mentions themselves. Various lexical and syntactic features have been extracted to describe the properties underlying these knowledge source, including word features, POS features, entity type, and several chunking features. All the adopted

knowledge is domain-independent.

Chapter 9 evaluates the utility of the features in the relation extraction task. By gradually increasing the feature set, we found that all of the four categories of features contributes to the relation extraction task more or less, hence, the incorporation of diverse features enables our system achieve the best reported performance. In addition, Evaluations in Chapter 8 and 9 also show us the influence of the setting of context window size, which indicate that extending the context too much may not improve the performance since the process may incorporate more noise information to confuse the characteristics of relation instance.

9.2 Future Work

In addition to the contributions made by this thesis, a number of further contributions can be made by extending this work in new directions. Some of these potential extensions are discussed below.

Our proposed semi-supervised and unsupervised methods are mostly feature based method, similarity between two relation instances are measured using the feature vectors derived from the context of two entity mentions. Firstly, since the feature space is relatively sparse, in order to improve the searching efficiency and to optimize the clustering result, in the future, we could apply some feature selection techniques to select an important feature set beforehand to construct context vectors (Roth and Lange, 2003). Secondly, as an alternative to the feature-based method, we have mentioned earlier that kernel-based methods (Zelenko et al., 2002; Culotta and Soresen, 2004) have the special property, that is, they are able to exploit non-local dependencies. Inspired by this, in the future, we could also consider to incorporate the tree

similarity function into our learning models so that we could capture more structure information from the parse tree for a relation instance. Dependency structures appear to be a reasonable alternative since they naturally model verbs and their arguments, which is how many relations can be seen. Thirdly, currently, we only extracted those lexical and syntactic features derived from contexts of entity pairs. We could investigate effective ways to explore semantic knowledge such as WordNet and namelists, to assist the relation extraction task.

For relation extraction problem, unsupervised learning solution is a promising topic of research. We can not expect that unsupervised methods will ever exceed supervised methods in cases where there is plenty of labeled training data, but we can hope that, when only unlabeled data is available, unsupervised methods will be important and useful tools. As described in our previous work on relation extraction (Chen et al., 2005a; Chen et al., 2005b; Chen et al., 2006b; Chen et al., 2006d), unsupervised learning method does not need a large amount of labeled data as their precondition, so it would make great significance if we can further improve the performance of our methods presented in this thesis. However, detecting relations is a difficult task for an unsupervised method because the set of all non-relation instances is extremely heterogeneous, and is therefore difficult to characterize them with a similarity metric. We believe that our work has made an importance in the right direction to lead to more future exciting work in unsupervised learning in automatic relation extraction.

Bibliography

E. Agichtein and L. Gravano. 2000. Snowball: Extracting Relations from large Plain-Text Collections. In *Proceedings of the* 5^{th} *ACM International Conference on Digital Libraries (ACMDL'00).*

M. Belkin and P. Niyogi. 2002. Using Manifold Structure for Partially Labeled Classification. *Advances in Neural Infomation Processing Systems 15.*

T. Berners-Lee, J. Hendler, and O. Lassila. 2001. The Semantic Web. *Scientific American.*

D. Bikel, R. Schwartz, and R. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning Journal Special Issue on Natural Language Learning.*

A. Blum and S. Chawla. 2001. Learning from Labeled and Unlabeled Data Using Graph Mincuts. In *Proceedings of the 18th International Conference on Machine Learning.*

A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *COLP: Proceedings of the Workshop on Computational Learning Theory.*

A. Blum, J. Lafferty, R.Rwebangira, and R. Reddy. 2004. Semi-Supervised Learning

Using Randomized Mincuts. In *Proceedings of the 21th International Conference on Machine Learning*.

Sergey Brin. 1998. Extracting patterns and relations from world wide web. In *Proceedings of WebDB Workshop at 6th International Conference on Extending Database Technology (WebDB'98)*.

E. Charniak. 1999. A Maximum-entropy-inspired parser. *Technical Report CS-99-12,Computer Science Department, Brown University*.

Jinxiu Chen, DongHong Ji, ChewLim Tan, and ZhengYu Niu. 2005a. Automatic Relation Extraction with Model Order Selection and Discriminative Label Identification. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*, Jeju Island, Korea.

Jinxiu Chen, DongHong Ji, ChewLim Tan, and ZhengYu Niu. 2005b. Unsupervised Feature Selection for Relation Extraction. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*, Jeju Island, Korea.

Jinxiu Chen, DongHong Ji, ChewLim Tan, and ZhengYu Niu. 2006a. Relation Extraction Using Label Propagation Based Semi-Supervised Learning. In *Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL-2006)*, Sydney, Australia.

Jinxiu Chen, DongHong Ji, ChewLim Tan, and ZhengYu Niu. 2006b. Unsupervised Relation Type Disambiguation Using Spectral Clustering. In *Proceedings of the joint conference of the International Committee on Computational Linguistics*

and the Association for Computational Linguistics (COLING/ACL-2006), Sydney, Australia.

Jinxiu Chen, DongHong Ji, ChewLim Tan, and ZhengYu Niu. 2006c. Semi-supervised Relation Extraction with Label Propagation. In *Proceedings of the Human Language Technology conference - North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL 2006)*, New York, USA.

Jinxiu Chen, DongHong Ji, ChewLim Tan, and ZhengYu Niu. 2006d. Unsupervised Relation Disambiguation with Order Identification Capabilities. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, Sydney, Australia.

M. Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the ACL1997.*

M. Collins. 2002. Discriminative Trainning Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of Empirical Methods in natural language Processing.*

A. Culotta and J. Soresen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of 42th Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

Y. Freund and R.E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning, 37(3):277-296, 1999.*

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Hongjun Lu. 2002. Discriminative Category Matching: efficient Text Classification for Huge Document Collections.

In *Proceedings of the IEEE International Confernece on Data Mining (ICDM)*, Maebashi City, Japan.

T. Hasegawa, S. Sekine, and R. Grishman. 2004. Discovering Relations among Named Entities from Large Corpora. In *Proceedings of 42th Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain.*

Minlie Huang, Xiaoyan Zhu, Donald G.Payan, Kunbin Qu, and Ming Li. 2004. Discovering patterns to extract protein-protein interactions from full biomedical texts. In *Proceedings of 20th International Conference on Computational Linguistics.*

Valentin Jijkoun, Jori Mur, and Maarten de Rijke. 2004. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of COLING-2004.*

T. Joachims. 2002. Learning to Classify Text Using Support Vector Machines. *Kluwer.*

N. Kambhatla. 2004. Combining lexical, syntactic and semantic features with Maximum Entropy Models for extracting relations. In *Proceedings of 42th Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

R. Kannan, S. Vempala, and A. Vetta. 2000. On clustering: Good,bad and spectral. In *Proceedings of the 41st Foundations of Computer Science, pages 367-380.*

Boris Katz and Jimmy Lin. 2003. Selectively Using Relations to Improve Precision in Question Answering. In *Proceedings of the EACL 2003 Workshop on Natural Language Processing for Question Answering*, Budapest, Hungary.

T. Lange, M. Braun, V. Roth, and J. M. Buhmann. 2002. Stability-Based Model Selection. *Advances in Neural Information Processing Systems 15.*

E. Levine and E. Domany. 2001. Resampling Method for Unsupervised Estimation of Cluster Validity. *Neural Computation, Vol.13, 2573-2593.*

J. Lin. 1991. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information Theory*, pages Vol 37, No.1, 145–150.

K. Litkowski. 1999. Question-answering using semantic relation triples. In *E. Voorhees & D. Harman, (red.), Proceedings of the eigth Text Retrieval Conference (TREC 8)*, Gaithersburg, Maryland.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics, 19(2):313-330.*

A. McCallum and D. Jensen. 2003. A Note on the Unification of Information Extraction and Data Mining using Conditional-Probability. In *Workshop on Learning Statistical Models from Relational Data at IJCAI'03.*

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

Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. 2005. Simple Algorithms for Complex Relation Extraction with Applications to Biomedical IE. In *Proceedings of ACL2005.*

S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proceedings of 6th Applied Natural Language Processing Conference*, Seattle USA.

Defense Advanced Research Projects Agency, 1995. *Proceedings of the sixth Message Understanding Conference (MUC-6).* Morgan Kaufmann Publishers, Inc.

Roberto Navigli and Paola Velardi. 2004. Learning Domain Ontologies from document Warehouses and Dedicated Web Sites. *Computational Linguistics, Vol. 30, Issue 2.*

A. Y. Ng, M. Jordan, and Y. Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Proceedings of Advances in Neural Information Processing Systems, pages 849-856.*

Zhengyu Niu, Donghong Ji, and Chew Lim Tan. 2004. Document Clustering Based on Cluster Validation. In *Proceedings of CIKM'04, Washington, DC, USA, November 8-13.*

Borys Omelayenko. Learning of Ontologies for the Web: the Analysis of Existent Approaches. In *Proceedings of the International Workshop on Web Dynamics, held in conj. with the 8th International Conference on Databased Theory (ICDT'01)*, London, UK.

Adwait Ratnaparkhi. 1999. Learning to Parse Natural Language with Maximum Entropy. *Machine Learning (Special Issue on Natural Language Learning), 34(1-3):151-176.*

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

Volker Roth and Tilman Lange. 2003. Feature Selection in Clustering Problems. In *NIPS2003 workshop.*

Gerard Salton. 1998. Automatic Text Processing: The transformation, analysis, and retrieval of information by computer. *Addison-Wesley.*

G. Sanguinetti, J. Laidler, and N. Lawrence. 2005. Automatic determination of the number of clusters using spectral algorithms. *IEEE Machine Learning for Signal Processing.*

D. Shen and D. Klakow. 2006. Exploring Correlation of Dependency Relation Paths for Answer Extraction. In *Proceedings of the ACL 2006.*

J. Shi and J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 22(8):888–905.

N. Slonim, N. Friedman, and N. Tishby. 2002. Unsupervised Document Classification Using Sequential Information Maximization. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

K. S. Tjong and M. F. De. 2003. Introduction to the CONLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CONLL-2003.*

V. Vapnik. 1998. Statistical Learning Theory. Whiley, Chichester, GB.

Yair Weiss. 1999. Segmentation using eigenvectors: A unifying view. *ICCV(2)*, pages pp.975–982.

D. Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.

D. Zelenko, C. Aone, and A. Richardella. 2002. Kernel Methods for Relation Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia.

H. Zha, C. Ding, M. Gu, X. He, and H. Simon. 2001. Spectral Relaxation for k-means clustering. *Neural Information Processing Systems (NIPS2001)*, pages 1057–1064.

Min Zhang, Jian su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. 2005. Discovering Relations Between Named Entities from a Large Raw Corpus Using Tree Similairty-based Clustering. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*, Jeju Island, Korea.

Zhu Zhang. 2004. Weakly-supervised relation classification for Information Extraction. In *Proceedings of ACM 13th conference on Information and Knowledge Management (CIKM'2004)*, Washington D.C.,USA.

GuoDong Zhou, Jian Su, Jie Zhang, and min Zhang. 2005. Exploring Various Knowledge in Relation Extraction. In *Proceedings of 43th Annual Meeting of the Association for Computational Linguistics*, USA.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from Labeled and Unlabeled Data with Label Propagation. *CMU CALD tech report CMU-CALD-02-107*.

Xiaojin Zhu, Zoubin Ghahramani, and J. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the 20th International Conference on Machine Learning*.