<table>
<thead>
<tr>
<th>Title</th>
<th>Entity-Centric Discourse Analysis and Its Applications (Dissertation 全文)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Wang, Xun</td>
</tr>
<tr>
<td>Citation</td>
<td>Kyoto University (京都大学)</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2017-11-24</td>
</tr>
<tr>
<td>URL</td>
<td><a href="https://doi.org/10.14989/doctor.k20777">https://doi.org/10.14989/doctor.k20777</a></td>
</tr>
<tr>
<td>Rights</td>
<td>The contents of chapter 3 were first published in CIKM'15. Copyright©ACM. DOI:<a href="http://dx.doi.org/10.1145/2806416.2806637">http://dx.doi.org/10.1145/2806416.2806637</a>.; The contents of chapter 4 were first published in NAACL-HTL'15. Copyright©ACL. DOI:<a href="http://dx.doi.org/10.3115/v1/N15-1030">http://dx.doi.org/10.3115/v1/N15-1030</a>.; The contents of chapter 5 were first published in COLING'16. Copyright©ICCL.</td>
</tr>
<tr>
<td>Type</td>
<td>Thesis or Dissertation</td>
</tr>
<tr>
<td>Textversion</td>
<td>ETD</td>
</tr>
</tbody>
</table>

Kyoto University
Entity-Centric Discourse Analysis and Its Applications

Xun WANG
Abstract

Computers have long been greatly facilitating our daily life with its massive storages and powerful calculation ability. Recently they even start to exhibit intelligence in a certain level by beating humans in some rather difficult tasks, like recognizing objects from images and videos, translating speech to text and playing chess. However, performing repeatedly operations or seeking optimized solutions with clearly defined constraints or rules is still far from complete intelligence. When it comes to circumstances where knowledge can not be explicitly or completely presented, computers are quite challenged.

Language for computers is particularly such a case. Processing natural language with intelligence similar to human beings remains a huge challenge for computers. It is one of the major problems for today’s artificial intelligence. Up till now substantive works has been conducted in this aspect, significant progress has also been made. But we are still far from achieving a perfect solution. Many problems regarding languages are AI-complete (Artificial Intelligence complete) which means if we can successfully resolve these problems, we would be able to build AI systems as intelligent as humans. The famous Turing Test, which is designed to evaluate a machine’s intelligence by talking with it, is exactly such an AI-complete problem.

Among the persistent efforts put by the research community, “discourse analysis” has drawn much attention. Discourse analysis is proposed to address various problems relating to building meaning from a piece of text. It aims to extract all the information carried by a piece of text and present the information in a computer-understandable way. This is a huge challenge as human beings’ understanding on language is still rather limited. Much of the inner deep mechanism
of human language remains unknown. Researchers have proposed various linguistic theories to interpret language phenomena. Among them, the most recognized one is structural linguistics. It provides a scheme for understanding language by analyzing its components. Language, like a building, is composed of different elements, of which the basis is morphemes. Morphemes form words, and words further compose phrases. One can use such elements, according to their demands and following certain rules and mechanisms, to generate sentences, paragraphs, and documents to express his/her intentions.

This structural point of view on text construction leads to an explanation about how information is encoded in discourse. When constructing discourse, small units like words and phrases, through mutual interactions, are assembled into large units like sentences and paragraphs. The interactions between text units play a vital role which bring forth the effect as “The whole is greater than the sum of its parts”. A larger text unit normally carries much more information than just a pile of its components. For example, a well-formed sentence like “I have a dog” carries more meaning than just an unordered set of words like “{dog, I, have, a, .}”. This sentence describes the fact that the subject “I” owns a “dog” and the relation between “I” and “a dog” is clearly defined as “possession” by the pattern “A have B”. While an unordered set of words “{dog, I, have, a, .}” does not have such a meaning. As is shown by this example, when parts join into a whole, there is always new information being generated.

The above analysis on the structure of discourse inspires a lot of work in NLP research. These work generally regards text as a whole and analyzes its components bottom-up to extract information and build meaning from a sequence of words.

But the mechanism of building meanings from smaller units to larger ones is not clear yet, which makes it hard to extract information from text. This work provides an alternative view of discourse. We regard discourse as a system constructed by entities, and explore several important problems emerging in the entity-centric analysis of discourse. We start our analysis from words and view

\[1\] Note that the construction of words from morphemes is quite special. We do not discuss morphemes in this work.
words in text as the basis in a hierarchical structure. We propose to learn word embeddings from a multi-layer structure so that the interactions between words far away from each other are taken into consideration. Such long distance relations are critical in understanding text comprehensively but are often overlooked. The learned embeddings are proved capable of capturing more information by our experiments.

We further explore deep information that is hidden beneath text but vital for discourse analysis. It comes into our notice that during the construction of text, words of certain types are sometimes omitted for simplicity. A thorough deep analysis of discourse requires us to restore these missing words. This work presents a neural network model with tree-based features to uncover these hidden components in text.

The above analysis enhances our understanding of discourse but only provides us with the basic components of discourse. Towards more intelligent NLP systems, we need to explore higher level structure to get more information from discourse. As the key contribution of this work, we propose the entity-centric discourse analysis.

Entities can be understood as anything that exist, either physically or not. In our analysis, we find that a piece of discourse, which possesses a complex structure, is always organized around certain entities. Discourse describes the states of entities it contains and their mutual relations. These entities provide the basis on which the discourse is developed and guide the narration. Again we use the simple sentence “I have a dog” as an example. It talks about the fact that “I” own “a dog”. From an entity-centric view, it describes the relationships between two entities “I” and “dog”. To understand this piece of text, we only need to monitor the states of the two entities and know about the relations between them. Using “I”, “dog” and the relation between them, we present information carried by text to computers explicitly so computers “understand” this sentence and can leverage the understanding for various applications. Discourse understanding thus can be achieved by collecting entities in text and their mutual relations.

In this work we further address three challenging NLP tasks: summarization, reading comprehension and machine translation, to leverage the understanding of
discourse obtained by entity-centric analysis.

The above-mentioned tasks require understanding of large text units, like sentences, paragraphs and even documents. It is a huge challenge for the prevalent data-driven methods. As we know, a major problem faced by data-driven discourse understanding is data sparsity, especially when comes to large text units. An intuitive solution is to employ compositionality, which studies how one builds meaning for a large text using its components. While the problem of compositionality is that as language is still far from being completely understood, a poor simulation of text composition, either using shallow features or employing sophisticated recurrent neural models, is hard to yield good performances.

The entity-centric view discourse analysis provides a novel way of building meaning from text. Entities are regarded as the skeleton of discourse. By monitoring entities, we can address many problems regarding the contents of discourse.

We build an entity graph to represent text and extract sentences to form coherent summaries. As for question answering, we use entity-based memory networks to represent the narrative lines in stories. Based on the distributed representations of entity states, we are able to respond to incoming queries with regards to the contents of the story. We further develop an entity-based representation method for text. The proposed model is applied in machine translation and proves effective.
Acknowledgements

First and foremost, my sincere appreciation is extended to Professor Sadao Kurohashi. The discussions with Professor Kurohashi inspired many ideas presented in this work. The research road paved by Professor Kurohashi provides a correct direction and a clear guideline towards the ultimate goal of complete understanding of human languages. Work in this thesis follows this guideline and discusses some basic problems and important applications in natural language processing and computational linguistics.

I would also like to thank Professor Toru Ishida and Professor Tatsuya Kawahara for serving on my dissertation committee. Without their patience and guidance, this dissertation would not have been possible.

My gratitude is also due to Associate Professor Daisuke Kawahara and Lecturer Tomohide Shibata. The discussions about research projects and the valuable help offered to me make part of this work possible. I would also like to express my thanks towards other members of Kurohashi-Kawahara Lab for the help and advice I received.

Particular thanks is recorded to my colleagues in NTT Laboratories, especially to my co-authors, Dr Masaaki Nagata, Dr Katsuhito Sudoh, Dr Tsutomu Hirao, Dr Masaaki Nishino, Dr Jun Suzuki and Mr Yasuhisa Yoshida for the encouragements and guides in both work and daily life.

Finally I would like to thank my beloved family for the unconditional love all these years and after. I would like to dedicate this dissertation to them.
Contents

Acknowledgments v

1 Introduction 1

2 Overview of Discourse Understanding 9
  2.1 Introduction 9
  2.2 Discourse Understanding within the Structuralism Paradigm 10
  2.3 Discourse Understanding within the Transformational-Generative Paradigm 15
  2.4 The Entity-Centric Approach to Discourse Understanding 17

3 Learning Word Embeddings 21
  3.1 Introduction 21
  3.2 Model 24
     3.2.1 Notation 24
     3.2.2 Revisit Distributed Neural Language Models 24
     3.2.3 Joint Embedding Training from Hierarchical Structure 25
     3.2.4 Details of Implementation 26
  3.3 Experimental Results 27
  3.4 Conclusion & Analysis 30

4 Empty Category Detection 31
  4.1 Introduction 31
  4.2 Proposed Method 32
     4.2.1 System Overview 34
5 Entity-Based Coherent Summarization
  5.1 Introduction ............................................ 47
  5.2 Centering Theory and Coherence Modelling ................. 50
  5.3 Modelling Summarization ................................ 51
    5.3.1 Problem Setup ..................................... 51
    5.3.2 Summarization Considering Global Coherence .......... 52
    5.3.3 Summarization Considering Local Coherence .......... 54
  5.4 Experiments & Analysis ................................... 55
    5.4.1 Experiment ......................................... 55
    5.4.2 Evaluation & Discussion ............................. 57
  5.5 Related Work ............................................ 59
    5.5.1 Salience-Focused Method ............................ 60
    5.5.2 Coherence-Focused Method ............................ 61
  5.6 Conclusion .............................................. 62

6 Entity-Based Memory Network ................................. 63
  6.1 Introduction ............................................ 63
  6.2 Memories in Deep Neural Networks .......................... 65
  6.3 Approaches .............................................. 67
    6.3.1 Overview ............................................ 67
    6.3.2 Entity-Based Memory Network Model .................... 69
    6.3.3 Implementation ...................................... 72
    6.3.4 Data Preparation .................................... 73
  6.4 Experiments .............................................. 74
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>An Example of Deep Information in Text</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Architecture of a Sentence</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Communication as Discourse Understanding</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Differences in Language</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>Text Units in a Hierarchical Structure</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>System Architecture of EC Detection</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Possible EC Locations</td>
<td>36</td>
</tr>
<tr>
<td>4.3</td>
<td>ECs in a Dependency Tree</td>
<td>37</td>
</tr>
<tr>
<td>4.4</td>
<td>EC Detection</td>
<td>39</td>
</tr>
<tr>
<td>4.5</td>
<td>An Example of Empty Category</td>
<td>43</td>
</tr>
<tr>
<td>5.1</td>
<td>A Complete Graph with Dummy Start and End Nodes</td>
<td>54</td>
</tr>
<tr>
<td>6.1</td>
<td>An Example of Question Answering</td>
<td>64</td>
</tr>
<tr>
<td>6.2</td>
<td>Comparison of the Memory Network and the Entity-Based Memory Network</td>
<td>66</td>
</tr>
<tr>
<td>6.3</td>
<td>An Example from bAbI</td>
<td>67</td>
</tr>
<tr>
<td>6.4</td>
<td>Architecture of the Entity-Based Memory Network</td>
<td>69</td>
</tr>
<tr>
<td>6.5</td>
<td>The Generalization Module</td>
<td>70</td>
</tr>
<tr>
<td>6.6</td>
<td>The Output Feature Module</td>
<td>71</td>
</tr>
<tr>
<td>7.1</td>
<td>An Example of the Entity-Based Machine Translation</td>
<td>84</td>
</tr>
<tr>
<td>7.2</td>
<td>The Entity-Based Encoding Module</td>
<td>86</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

7.3 The Sequence to Sequence Machine Translation . . . . . . . . . . . . . . 88
7.4 The Entity to Entity Machine Translation Model . . . . . . . . . . . . . . 89
List of Tables

3.1 Results on Word Similarity Task ........................................ 28
3.2 Results on Word Analogy Task ........................................ 29

4.1 Data Division and EC Distribution ...................................... 40
4.2 Parameter Tuning ............................................................. 40
4.3 EC Distribution in the Test Data ......................................... 42
4.4 Performance on the CTB Test Data ...................................... 42
4.5 Effectiveness of Features .................................................. 44

5.1 Human Evaluation on DUC2004 ........................................ 58
5.2 ROUGE Results on DUC2004 .............................................. 59

6.1 Results on Machine Comprehension Test Dataset .................. 76
6.2 Results on Large Movie Dataset ........................................ 77

7.1 Results on En-Ja Legal Documents Dataset .......................... 93
Chapter 1

Introduction

In the age of information overload, the requirement for intelligence for modern natural language processing (NLP) applications has becoming increasingly urgent. The traditional way of computer-human interaction cannot provide users with the desired information from the massive data stored in computers efficiently. Neither locating useful information from hundreds of thousands of web pages returned by search engines, nor looking for interesting news from a huge collection of news feeds is a pleasant experience. Users hope computers can exert the ability of intelligence by understanding the intention of humans and providing the needed information automatically, no more, no less.

Driven by the demand of understanding text to cater various requirements, the task of discourse understanding has been intensely studied. Discourse understanding aims to enable computers to build meaning from a segment of text. It is a huge challenge and has drawn great interests from the NLP community.

Work in this field can be roughly classified into the following categories according to the methods adopted. One kind is referred to as rule-based methods. This kind of methods dates back to the early days of discourse understanding. They assume that language is constructed using a vocabulary conforming to a coherent set of grammar rules. Parsing a piece of text is similar to, if not exactly the same as, parsing a segment of computer codes. Owning to the continuous efforts of the NLP community, the rule-based methods reach their bottlenecks in a short time as the number of rules needed to model language increases explosively. It is not
long before it becomes impossible to design a coherent system which is able to
describe the complex linguistic phenomena. Nowadays, the predominant meth-
ods are statistical model-based. Statistic models are trained using datasets in
expectation of that the designed model could capture certain linguistic patterns.
Advances in natural language processing tasks depend heavily on technologies of
machine learning as we have witnessed in recent years, like the Support Vector
Machine (SVM) model for classification tasks, the topic models for clustering as
well as the recent boom of the neural networks for many applications. The rapid
development of machine learning models helps deepen our understanding of lan-
guages which in turn assists us in designing better models for NLP tasks. The
mutual reinforcement between NLP tasks and machine learning models greatly
promotes our study of languages and benefits NLP applications.

Along with the research on discourse understanding goes deep, new problems
emerge. Languages in essence are not the same as other data forms like figures
that machine learning models deal with. The difference mainly lies in that lan-
guage possesses complex structures and intricate inner mechanism. Linguists have
proposed various theories to interpret different language phenomena. These lin-
guistic insights have been widely used to develop suitable models for NLP. In fact,
for NLP, a good theory usually leads to methods that work well in practice and
vice versa.

As for discourse analysis, the structural linguistics is widely adopted. From
a structural linguistics point of view, discourse is assembled by its components.
The information contained in text is determined by its components and their mu-
tual relations. Thus to identify the text components and analyze their mutual
relations have long been the research focus of most NLP work. This work follows
this guideline and explores several important problems in discourse understand-
ing using machine learning approaches. We leverage linguistic insights to design
models with respect to the structure of languages and test the proposed models
on practical applications. Through a variety of experiments, our models have
proven effectiveness in several NLP applications, which is also of great help for
understanding the inner mechanism of language.

Some information contained in text is explicit and easy to extract but some
天気がいい。友たちと公園へ行きます。

1. The date is 今日.
2. The topic is 私.
3. “天気がいい” is the reason and “公園へ行きます” is the result.
4. It is highly probable that the emotional mood of the subject is happy.
5. It is also highly probable that “私” is walking in the park now.

Figure 1.1: An Example of Deep Information in Text

is not. Explicit information, like bag of words, is presented clearly. But more information is buried beneath the text. If we call what is explicitly presented before us or can be easily obtained as “shallow information”, those hidden in text can be referred to as “deep information”.

Fig. 1.1 serves as an example about deep information in discourse. It shows what is obvious to humans but hard to identify for computers. Exploring such information is vital for an intelligent NLP system, but more often than not it is buried deeply in the text and can hardly be recovered unless we conduct a deep analysis of text as is shown in Fig. 1.2. In deep analysis, we regard text as a hierarchical structure constructed bottom-up. And to explore the interactions of components in the same layer and those between adjacent layers are crucial for understanding text.

In this work, we propose methods for discourse analysis to unveil hidden or deep information, and design novel models to leverage the acquired deep information towards intelligent NLP systems. We develop a novel method for learning embeddings from hierarchical structure which considers the long-distance relationships between words. The learnt word vectors thus can carry more information. As high quality word representations are the basis of many deep learning NLP models, the learnt word embeddings from our model provide a solid foundation for our research. We then focus on discovering the hidden nouns in sentences. As
we have stated, knowing the components of text is a vital step of understanding text. As the example in Fig. 1.2 shows, some words are omitted and need to be recovered, so they are explicitly presented to computers. We propose to use tree features to detect these hidden parts in text.

These deep information we have acquired via discourse analysis can be incorporated in down-stream applications in various ways. One of the simplest ways is to use such information as features and pass them to existing models to achieve better results. But as stated before, with a deeper understanding of text, we can leverage our knowledge of discourse to design models with regard to the inner mechanism of language. In our analysis, we find that discourse is elaborated around the entities it mentions. At each level of text, the entities are always the cores and the remaining parts either describe the states of entities or explain the relations between them as is shown in Fig. 1.2. The complex structure of text can thus be projected to a world of entities with the help of deep discourse analysis. In this work, we develop novel machine learning methods leveraging the deep information obtained using entities for practical applications. We address three important NLP problems, automatic summarization, reading comprehension and machine translation. These problems serve different purposes. Nonetheless, they all require a deep understanding of discourse.

This work can be roughly divided into two parts. The first part is about identifying the deep information carried by text using deep discourse analysis. The second part is about leverage the deep information to develop systems for practical applications based on understanding of text.

The dissertation is organized as follows:

Chapter 2 gives a brief introduction to discourse understanding. We summarize some important linguistic theories including structuralism and generative grammar. These theories reflect our understanding of discourse. Based on the limited knowledge about discourse structure, we propose the entity-centric method for discourse understanding and use it to resolve various problems. Our work in return helps deepen our understanding of discourse.

Chapter 3 introduces how to learn better word embeddings by taking the discourse structure into consideration. Usually the embeddings of words are learnt
Note we omit morphemes in this figure since morphemes are widely believed to behave in a different way.

according to their neighbouring words. The long-distance relations and the structure of text are left out. We propose to model the text as a hierarchical structure which consists of several layers interacting with each other. Words in a document are all connected via their common ancestors. The embeddings of words and components of other layers are learnt simultaneously. Experiments on several tasks show that the learnt embeddings are more powerful in representing meanings of words than those learnt without considering the text structure.

Chapter 4 presents a method to detect the hidden components in discourse. In discourse analysis, often is the case that we have to consider hidden information which is not expressed explicitly but carried by the context. One example is the empty category phenomena. Prop-drop languages, such as Chinese and Japanese, tend to omit nouns and pronouns when they are inferable from the context. These missing nominal words are referred to as empty categories. We propose a novel empty category detection method which represents various tree-based features using vectors and employs deep neural networks to detect the missing words.

Chapter 5, Chapter 6 and Chapter 7 are related to improving performances of practical applications. Discourse is about passing information by text correctly
CHAPTER 1. INTRODUCTION

and completely to the recipients. From the view of structural linguistics, it is until we decipher signifier into signified that discourse ends. In communication, the recipient usually gives appropriate response showing that the signified have been understood. As for discourse understanding, we can only tell if the model understands the discourse or not according to its response, i.e., whether it gives the desired output or not. Some work on discourse analysis extract information carried by text and presents it in certain forms like parse trees or relations. Both trees and relations have been widely studied and proven useful in downstream tasks. But in this work, instead of using traditional representational forms, we turn to entities and relations to represent information contained in discourse. To verify the effectiveness of the proposed discourse analysis method, we design models that work with entities for practical applications. In fact, these models are also parts of the understanding since we have not turned text into signified. The information about signified is encoded inside entities and only with the proposed model can computers read out the encoded information and response correctly.

In Chapter 5, we present a model based on entities to generate coherent summaries. We find entities play a key role in maintaining coherence and carrying information in text. The proposed model represents text by entities and adopts a novel objective function based on entities. The objective function considers informativeness and coherence simultaneously. The problem of extractive summarization is formulated as an integer linear programming problem. By working on entities, the proposed method does not require sophisticated analysis and can be applied to many languages easily.

In Chapter 6, we present a comprehension-based multi-purpose question answering system which to some extent can answer almost all questions in natural language processing. A correct and thorough analysis of text should enable computers to answer any question raised with regard to the text. Instead of training different ad-hoc models for different tasks, we present a unified model which is capable of solving a number of natural language processing tasks. In the hierarchical analysis of text, the meaning of text lies in the components it contains and the structure constructed by the components. We design an entity-based model which use entities to model the text. By analyzing the states of entities, we are able to
respond to questions raised with regard to the text. Experiments are conducted on several natural language processing tasks by using the proposed model and report satisfying results. As an approximation of text architecture, the proposed model is able to capture relations between entities and answer different questions accordingly. Normally highly differentiated models and features are required for different tasks, but here we use a unified system with unified features. It is one step further on the road to complete artificial intelligence.

Chapter 7 is about adopting the entity-based text representation in machine translation task. We develop a novel model using entities to represent text in machine translation. Existing neural networks represent one sentence as one single vector which we believe is not sufficient. We decompose sentences into entities and their mutual relations. For each sentence, we use several vectors to represent it. These vectors correspond to entities and relations separately. This model is especially useful for machine translation task as the correspondence between entities of different languages is easy to model. We propose a three-step fully supervised learning model to translate sentences.

Chapter 8 concludes this thesis and summarize the research achievements of this work. We dive into the discourse and try to find the discourse structure beneath text. Improvements on several tasks verify the correctness of the research guidelines we follow.
Chapter 2

Overview of Discourse Understanding

2.1 Introduction

The term “discourse” is widely used in natural language processing and many other fields. In different context, the word “discourse” emphasizes different aspects. But nonetheless it refers to the communication between speakers and listeners/audience. Human beings form ideas in their minds through interactions with out world. Discourse starts when one tries to pass one idea to another human being. The idea is firstly represented in the form of language and then passed to another human being in the form of sounds or written symbols. The receiver (listener or reader) interprets the received sound or symbols and forms an idea in his/her mind. Discourse ends when an idea is conveyed from one to another.

According to Ferdinand de Saussure [22], the idea, which is formed inside one’s mind is the signified element while the acoustic signs or written symbols are the signifying element. Together they construct language. Take Fig. 2.1 as an example. One comes up with the idea of “go to the park” in one’s mind. The purely conceptual mass of the idea is what one wants to talk about, the signified part. One may use “Go to the park” or “去公园” or “公園へ行く” or other languages to express this idea. The concrete signs, either acoustic or written forms, are referred to as “the signifying part”.

Human beings have the ability of using languages to express ideas and interpreting languages to receive ideas from others, in other words, bridging the gap between signifier and signified. The discourse analysis, deals with the signifying and tries to understand what it signifies. It tries to replicate human ability of using language for computers. As is widely believed, an idea can only exist in the form of language. If computers possess the ability of understanding language, it will be able to understand any ideas we have, which leads to complete artificial intelligence. In this sense, discourse understanding is AI-complete.

However, currently building such a system is far beyond our capability. A lot of theories have been proposed to interpret the mechanism of human language. Among them the structural linguistics and generative grammar have gained the most popularity.

2.2 Discourse Understanding within the Structuralism Paradigm

Structural linguistics was first developed by Ferdinand de Saussure and enjoyed rapid development, especially after his book *The Course in General Linguistics* edited by his students from his lecture notes was published. Nowadays the struc-
2.2. DISCOURSE UNDERSTANDING WITHIN THE STRUCTURALISM PARADIGM

tural linguistics has many sub-branches and contains distant even controversial hypothesis, explanations and theories. Linguistics is deeply impacted by structural linguistics. In fact the impact of structural linguistics has gone beyond linguistics and shown influence in many other fields. The word “structuralism” and the idea of defining one item according to its relations to other items have been widely adopted in different fields, like literary, psychology, sociology, anthropology and so on. Saussure’s view of language can be briefly categorized into several aspects:

- Sign: The sign is the most basic element of language. A sign contains two inseparable parts, the signifier and the signified. As the example shown in previous section illustrates, the signifier is something we use to refer to another thing. For example, pupils always use fingers when doing addition and subtraction like “Given 5 apples, how many are left if I ate 3”. In this case, fingers are the signifier and apples are what the signifier refers to, the signified. In the case of pupils, the relation between signifier and signified is temporally. After the calculation, fingers do not represent apples any more. When it comes to language, the relation is much more persistent and widely recognized. However, one thing remains unchanged: the correspondence between the signifier and the signified is arbitrary.

- System: Since the relation between the signifier and the signified is arbitrary, how do we make meaning from the signifier? The answer given by Saussure is the language system. There exists a language system which arranges the elements it contains. The system defines what elements it contains and how one element relates to others. This system makes it possible for us to say anything we want to. In this sense, the system equals to the language ability and is what discourse understanding aims to simulate. However, the system, as is argued by Saussure, is implicit. What we observed are the utterances. Anything we say is an instance of the language system. An existing system governs everything about language including what we have said, what we are saying and what we will say. The invisibility of language system becomes a huge challenge for linguists. One can only study the language system by
monitoring its utterances.

- Difference: If the language is a system well organized as Saussure believed, we should be able to analyze the system by decomposing it into its elements and describe each element by its relations to the remaining. System is a sum of its elements and the relations between them. However, it is impossible to describe the basic element in language, sign, clearly and unambiguously. “A language is a system of differences with no positive terms.” Saussure pointed out that we distinguish one signifier from another because of the differences between them. For example, if the word “dog” is used the same with “cat”, we would not be able distinguish between the two words. But in practice, we use “dog” to refer the animal which carefully and loyally guards our houses and “cat” for the animal which sleeps all day long. “Dog” and “cat” as two words do not make sense but the differences between them and other words matter. Here comes another problem. A huge difference distinguishes two signs. For example, the differences between the concept of “dog” and that of “cat” are so huge that we have to use two signs to tell one from the other. But we do not create two separate words for a two-month dog and a three-month dog, at least for English, Japanese and Chinese. Of course, if people realize the differences which are often ignored in other societies, their language systems usually possess unique signs to address such differences. Fig. 2.2 shows an example about the differences in language. It is quoted from Classic of Poetry, (诗经·鲁颂). This poem talks about horses in the field, to be specific, male horses. Male horses can be further divided into subtypes according to certain standards. Some of these differences matter so much that in English some unique words are used to address these differences.

- Stallion: a fully grown male horse, especially one that is used for breeding.
- Gelding: a horse that has been castrated.
- Colt: a young male horse up to the age of four or five.
- Sire: the male parent of an animal, especially a horse.
2.2. DISCOURSE UNDERSTANDING WITHIN THE STRUCTURALISM PARADIGM

駉駉牡马，在坰之野。薄言駉者，有騥有皇，有骊有黄，以车彭彭。思无疆，思马斯臧。
駉駉牡马，在坰之野。薄言駉者，有骓有駓，有骍有骐，以车伾伾。思无期，思马斯才。
駉駉牡马，在坰之野。溥言駉者，有駒有駱，有骝有雒，以车绎绎。思无斁，思马斯作。
駉駉牡马，在坰之野。溥言駉者，有骃有騢，有驔有鱼，以车祛祛。思无邪，思马斯徂。

Figure 2.2: Differences in Language

But in ancient Chinese, people have different focuses.

- 験: black horses with white butock.
- 皇: “騩” yellow-white mixed horses.
- 骊: black horses.
- 黄: brown horses.
- 靛: black-white dappled horses.
- 驘: red horses.
- 驤: dark-blue horses.
- 驪: dark-blue horses with squamous stripes.
- 驴: red-body, black-bristle horses.
- 鴴: black-body, white-bristle horses.
- 騴: light-black white dappled horses.
- 騛: red-white dappled horses.
- 騥: black-body, yellow-back horses.
- 鱼: horses with white furs around eyes.
This example clearly demonstrates how important difference is for language. If we failed to notice the differences, there would be no associated signs. On the other hands, we would not be able to express the differences without the help of different signs. But differences are obviously different in different language systems. As demonstrated above, colour of horses is very important in ancient Chinese but not as important in modern English.

The idea of difference leads to another important term “to a certain extent”. If “A” differs from “B” a little, and the difference is not so huge or at least not so huge for us, we will regard them as the same. Signs, defined by differences, fluctuate within a range allowed by the language system. As we know, computers tend to be accurate and prefer a fixed representation of an item. The flexibility and ambiguity of language makes it even more challenging for computers to understand language.

Structural linguistics is one of the most influential theories regarding the nature of language. To a certain extent, all modern linguistics is structural. It is the same with natural language processing. Structural linguistics now becomes a large family with varied doctrines. Among them, Leonard Bloomfield [8] did an excellent work by a continuous attempt to define rigorous and precise systems for language. As we have stated, the key of analyzing a system is to define its components and describe the relations between them. Bloomfield proposed a series of postulates which he believed “Nevertheless, ... further the study of language, because it forces us to state explicitly whatever we assume, to define our terms, and to decide what things may exist independently and what things are interdependent” [8]. These postulates focus on positions in text, like what kind of elements appears in which position. They are proposed for focusing on the system itself and peeling off meanings.
2.3 Discourse Understanding within the Transformational-Generative Paradigm

Despite its popularity, the structural linguistics is challenged by the generative linguistics (generative grammar). Noam Chomsky introduced the transformational generative grammar [15]. The term “grammar” is defined as the inner language ability of humans. This is clearly different from the system described by Saussure. The generative grammar attracted many researchers and began to develop quickly. Now it is also a collection of theories relating to each other. Some important concepts need to be addressed with regards to generative grammar.

- Universal Grammar: Humans, as is argued, gain their language ability, at least part of it, innately. There exists a universal grammar which applies to all human languages and this grammar is not acquired but planted inside our brains. Saussure pointed out that language phenomena are instances of language systems. And Chomsky goes one step further by arguing that language systems are in fact instances of a universal grammar. In the context of generative grammar, one language differs from another only in parameters. The learning of a language equals to learning its “parameters”. As we know, a sentence usually is constructed by “S”, “V” and “O”. This rule can be regarded as the universal grammar. But the orders in different languages are different. In English and Chinese, it is “S+V+O” and in Japanese it is “S+O+V”. English, Chinese and Japanese follow the same universal grammar by constructing sentences from “SVO” but use different parameters which control the position of “V”. The procedure of studying a language involves learning such parameters.

- Deep structures and surface structures: The two terms are used in the transformational grammar. Surface structures are what we observed and deep structures are what controls the surface structures. Beneath every sentence, there exists such a deep discourse that defines the relations between elements. It is just like marionette, surface structures are manipulated from beneath text by strings attached to a deep structure. The string here is
called transformational rules. From the view of generative grammar, a sentence, which is a surface structure, comes from a deep structure which defines its meaning. The transformation from deep structure to surface structure follows certain rules: transformational rules. In Saussure’s system, the language system’s generative ability results from the relations between elements and the fluctuations in meaning. Here it is due to the deep structures and transformation rules. The number of deep structures is much smaller than that of surface structures thus it is easy to store the knowledge about deep structures in human brains in advance.

As we know, structures and relations which are defined by structural linguistics have become the basis of natural language processing now. Meanwhile, generative grammar also exert great impact. A lot of work is based on generative grammar. Among them this work addresses the problem of “empty category”. Empty categories or zero-nouns is related to transformation from deep structures to surface structures. During the transformation, some elements are moved away from their original position or omitted (or lost their phonological content). But for discourse understanding, it is necessary to recover such elements as they are part of the deep structure which defines the meaning.

Transformational grammar aims to uncover the essence of language and provides novel insights into the underlying psychological mechanisms. To precisely describe deep structures and the transformation rules, several kinds of theories are proposed. However the theories about transformational grammar became complex when matured. Chomsky argued that a universal grammar should be simple and elegant so that it could be stored inside human brains.

- Minimalist Program: minimalist program is named as a “program” instead of a “theory”. It is an attempt to limit the number of operations and make the grammars as simple as possible. It is argued that the universal grammar is constructed under the principle that it is prefect. This means the universal grammar is highly optimized and possesses a design which is just
good enough with no room for improvements. It is also constructed under
the principle of economy which means in the universal grammar, nothing is
redundant.

Generative grammar provides novel insights about language. It assumes that
there exist a deep structure that controls the surface structure. The nature of
language could not be fully explored if we ignore deep structures which are hidden
beneath text.

2.4 The Entity-Centric Approach to Discourse Understanding

Not only the structural linguistics and the generative grammar are developing
rapidly, but also other theories which do not fall into the two kinds emerge. These
new hypothesis and theories cast novel insights to the nature of the underlying
human languages. Discourse understanding follows these guidelines and various
methods are developed to resolve practical applications. From a structural lin-
guistics view, we need to study the compositionality of discourse and know about
how elements of discourse are organized to construct meaning. Existing work gen-
erally follows this principle. Research on discourse starts from words and moves
to phrases, sentences, paragraphs and documents. These work faces at least two
problems. One is the data sparsity, and one more thing is that no theories ex-
plaining compositionality haven been working smoothly. It causes troubles for
discourse understanding. When analyzing large text units, researchers seem to be
at a loss and tend to regard text as simple data using surface features to train
statistical models without considering the complex systems or deep structures
beneath text.

Recently the neural methods which draw much attention further put forward
this tendency. The neural models originate in simulation of human brains and have
been proved successful in a series of tasks. These models generally ignore the inner
mechanism of language and use deep networks to extract features from training
data for specific tasks. Practical applications benefit greatly from neural models.
But there is still little progress in understanding the mechanism of language. This
work, on the other hand, discusses both aspects. We present an entity-centric view of discourse. Discourse, in this work, is regarded as the sum of its entities and their mutual relations. We also design models leveraging our knowledge of discourse for practical applications and conduct experiments to verify the effectiveness of these models.

The proposed entity-centric method fully explore existing theories on linguistics. As stated, discourse talks about ideas formed in one’s mind. The ideas inside our brains always elaborate around certain entities which exist physically or not. By monitoring these entities, we can extract information in discourse. Thus we need to collect all the entities in discourse. According to the generative grammar, some entities are not explicit in surface structures. These entities need to be recovered from text and presented to the computers explicitly. We develop a novel method to identify the zero nouns in discourse.

Note that here we focus on entities and ignore other parts in discourse. Information carried by other parts needs to be passed to entities. We turn to distributed representations for help. We use low dense vectors to represent entity states. The distributed representations help us quantify the differences, not just differences between different entities, but also that between different states of the same entity. Again we take the sentence “I have a dog” as an example. “I” and “dog”, as two signs refer to the concept of “I” and “dog” (the signified). But when the two words are used in the sentence “I have a dog”, their meaning changes slightly. The “I” in this sentence is not just the first person pronoun, but also the one who have a dog. And the dog here is not an arbitrary dog. It is the one which is owned by “I”.

From either the view of structural linguistics or the view of generative grammar, the differences between the concept “I”, the first person pronoun, and the entity “I” who actually owns a dog is due to the word’s relations with other parts. In this case, it is with “have a dog”. In our work, we only keep entities and use entities to carry all the information. Using distributed representations, entities are represented as vectors. Assume $\text{vec}(I) = \{v_0, v_1, ..., v_n\}$ which is learnt from a large corpus. This vector represents the concept “I”. As for the “I” in the sentence, we use a new vector $\text{vec}'(I) = \{v'_0, v'_1, ..., v'_n\}$. This vector differs from
2.4. THE ENTITY-CENTRIC APPROACH TO DISCOURSE UNDERSTANDING

In the context of discourse understanding, the vector $\text{vec}(I)$ and the difference between two “I” is explicitly expressed without referring to other parts. It is similar with the entity “dog”.

Above we elaborate the key idea of this work, encoding information of discourse into entities. By differentiating the general term of an entity and the specific entity that is talking about in the discourse, we reduce the number of elements that need to be dealt with. The proposed entity-centric view presents a new structure of discourse which is built on entities. To fully explore the entity structure, we design different models for different tasks. In this work, we address the problem of summarization, reading comprehension and machine translation. For summarization, we focus on maintaining the coherence when extracting important sentences. An entity-oriented graph is used to represent source documents and summaries are extracted as paths in the entity graph. As entities play an important role in maintaining coherence in discourse, the proposed method successfully improves the quality of generated summaries. When comes to reading comprehension and translation, as we said, all the information is encoded in entity vectors. We use the entity vectors for answering questions and doing translation. The details of our analysis and applications are shown in the following chapters.
Chapter 3

Learning Word Embeddings

3.1 Introduction

A fundamental problem in NLP is about representing words. One-hot representation uses a vector with size equals to that of the vocabulary to distinguish one word from others. The one-hot vector is an all-zero vector except only one position filled by one to indicate the word it represents. The disadvantages are obvious mainly in two ways, firstly, the one-hot representation requires numerous parameters to store one word; secondly, it cannot encode word relations as in one-hot representations, distances between any two words are equal. From the view of structural linguistics, one word should be defined by its relations with other parts in text. One-hot representation fails in the aspect as distances between all words are equal. The distributed representation [40] remedies this problem[4] by using real-valued, abstract and condensed vectors. The dimensions of word embeddings are usually only about several hundreds or even smaller. Also distances between word pairs are no longer equal. Words with similar meanings can be clustered together by measuring the Euclidean distances between their vector representations. All these provide great benefits for natural language processing tasks. It thus draws much attention to learn high quality word representations.

There are two main families for learning embeddings for words in the literature. The first family leverages document-level word-occurrence statistics, such as LDA [7], GloVe [47], or matrix factorization based approaches (e.g., LSA and SVD),
given the intuition that co-occurrent words are relevant. Such global co-occurrence statistics based models neglect word order information about how local meanings are formed by neighbouring words. The second family refers to local context window approaches (e.g., [5, 19, 73]). The downside of such models is that they poorly harness the global information at document, paragraph or sentence level. There are also some attempts trying to bridge the gap between the two families: [44] proposes a document-level vector leveraged from tf-idf into local learning process; paragraph vector [58] makes word prediction with the help of the leveraged document/paragraph/sentence level information; [61] explores hierarchical auto-encoder for paragraph and document representations. Their efforts prove useful for learning sentence, paragraph and document representations. While neither of them incorporates the relations between different levels into their methods to improve the word embeddings.

Towards better word embeddings, we take the intrinsic structure of text, about how units are arranged to form meaningful context, into full consideration.

(1) Horizontally: According to the discourse theory in early days (Mann and Thompson (1988)), in a coherent text, not only words, but clauses, sentences, and larger multi-clause groupings are tightly connected. Text units take their respective roles and interact with units at the same level (token-to-token, sentence-to-sentence and paragraph to paragraph) semantically, syntactically, and logically.

(2) Vertically: Words form the meanings of sentences, sentences form paragraphs, and then paragraphs form documents, which organizes the arrangements into a tree structure vertically.

The importance of tree structures for sentence, paragraph and document representations has been explored in previous research [100, 58]. In particular, the paragraph vector [58] use a two layer structure to model the relations between paragraphs and sentences. In this work, we further extend it to multi-layers to improve the word embeddings.

The proposed model captures the two aforementioned aspects in a unified embedding learning framework which holds promise to bridge the gap between the co-occurrence based and prediction based embedding learning frameworks. Horizontally, we model each level of units based on the Markovian manner, where
neighbouring units are correlated based on the similar assumption we make in language model. Vertically, each unit (e.g., sentence) exerts its impact on its subsidiary lower-level text units (e.g., words). Unlike [44] where document-level information is harshly incorporated, the proposed approach gently incorporates the order information at paragraph level and sentence level, and therefore preserves the semantic integrity of the contexts.

The adopted type of architecture arranges all text units in a unified structure, where influence of one unit is propagated to others (the siblings and children), naturally bridging the gap of and taking the merits from the aforementioned two learning families. To note, our approach is inspired by the paragraph vector model [58] which models paragraph and tokens within it in a two-level hierarchy, where words are predicted given neighbours and its resided paragraph.

The proposed architecture is ultimately grounded on the lowest level of the hierarchical structure, words, by predicting the current token, where the embeddings for neighbouring tokens, and higher levels text units are simultaneously updated. The proposed algorithm is a general one and can be adopted to currently prevailed frameworks, e.g., skip-gram models, CBOW and recurrent neural models [74]. The system can be optimized by standard strategies taken in embedding learning literature, either through standard softmax, or others like hierarchical softmax, or negative sampling.

Note that the proposed algorithm ends up with distributed representations for documents, sentences and words, which could be used as input for different applications and for different levels of units. But here we just keep the word embeddings as the proposed model has not been specially optimized with regard to upper level text units representations. For these large text units, it is still not clear how their meanings are made up from words they contain. Though several composition-based neural network models [103, 127, 100] have been proposed and proved useful in a range of tasks, none of them manage to achieve the expected level of performance as word embedding models do.

We evaluate the learning frameworks on word analogy and word similarity tasks, which are two basic tasks for word embedding evaluation. Experimental results demonstrate that by utilizing the hierarchical structure of documents, we
obtain better performances.

3.2 Model

We present our embedding learning model in this section.

3.2.1 Notation

Document $D$ is composed of a sequence of paragraphs $D = \{P_1, P_2, ..., P_{N_D}\}$, paragraph $P$ is composed of a sequence of sentences $P = \{S_1, S_2, ..., S_{N_P}\}$ and sentence $S$ is composed of a sequence of words $S = \{w_1, w_2, ..., w_{N_S}\}$, where $N_D$, $N_P$ and $N_S$ respectively denote the number of correspondent children in the document, paragraph and sentence. Each level text unit $D$, $P$, $S$, $w$ is associated with a $K$ dimensional embedding $e_D$, $e_P$, $e_S$ and $e_w$. All text units are therefore arranged into a tree hierarchy with $L = 4$ levels. Let $\eta$ denote any node in the tree, where $\eta$ could be document, paragraph, sentence or word with embedding $e_\eta$, parent($\eta$), and sibling($\eta$) respectively denote the parent and siblings of $\eta$.

3.2.2 Revisit Distributed Neural Language Models

We first briefly describe the general neural learning framework for word embedding widely adopted in the literature, the key idea of which is to optimize the framework by minimizing the conditional probability of predicting current word given preceding ones or neighbouring ones.

Consider a sequence of word tokens $\{w_1, w_2, ..., w_N\}$. The conditional probability of current word is given by:

$$
p(w_n|w_{n-1}, ..., w_{n-k}, w_{n+1}, ..., w_{n+k}) = f(e_n|g(e_{n-1}, ..., e_{n-k}, e_{n+1}, ..., e_{n+k}), \Theta)
$$

(3.1)

where $\Theta$ denotes the parameter space involved in the probability function $f()$. $g()$ denotes the operation performed on neighbouring vectors. Many types of $g()$ have been explored such as averaging neighbouring embeddings (CBOW) [73], getting the dot product between $w_n$ and each of its neighbours (skip-grams) [73],
concatenating neighbouring vectors and projecting the concatenation into a low-dimensional space (e.g., [20, 106]), or convolving the preceding words using a recurrent network [75].

Commonly used forms of \( f() \) include predicting current word using a softmax function or contrastive sampling. Many alternatives have been proposed for easy training, such as hierarchical softmax [76] and Noise-contrastive Estimation [106, 79].

### 3.2.3 Joint Embedding Training from Hierarchical Structure

Our model take advantages of the hierarchical structure of text.

- Horizontally, we incorporate Markov property at each level of the tree structure.
- Vertically, child embeddings are influenced by their parent nodes.

The model extends standard embedding learning framework by subsequently predicting embedding of every node \( \eta \) along the tree structure given its parent and siblings:

\[
p(\eta | \text{parent}(\eta), \text{Sibling}(\eta)) = f(e_\eta | g(e_{\text{parent}(\eta)}, \{e_{\eta'}, \eta' \in \text{Sibling}(\eta)\}, \Theta)) \tag{3.2}
\]
Thus, the probability of the whole document is given by:

\[
p(D|\Theta, e_D, \{e_P\}, \{e_S\}, \{e_w\}) = \prod_{\eta \in \text{Tree}} f(e_\eta|g(e_{\text{parent}(\eta)}), \{e_{\eta'}, \eta' \in \text{sibling}(\eta)\}, \Theta)
\]  

(3.3)

As can be seen, for two words that do not reside in the same sentence, they will still distantly interact with each other as the influence is propagated up to the sentence embeddings, paragraph embeddings and document embeddings, and then down to the other word. Therefore, the proposed model can to some extent capture global level statistics without losing the advantages of local neural composition.

On the other hand, based on the Markov property along each level of the trees, the meanings of adjacent text units interact with each other and preserves the integrity of meanings at each level, potentially leading to better representations at lower levels. Eventually these merits will be further propagated to word level prediction, leading to better word level embeddings.

For illustration purpose, we assume \( g() \) takes the form the concatenation of sibling embeddings and parent embedding. \( f() \) takes the form of sigmoid function at sentence/paragraph level and softmax at word level. Let \( P \) denote the paragraph that sentence \( S_i \) resides in, and \( S \) denotes the sentence that word \( w_i \) resides in, we have:

\[
p(e_{S_i}|\cdot) = \sigma(e_{S_i} \cdot g(e_P, e_{S_{i-1}}, \ldots, e_{S_{i-N}}))
\]

\[
p(e_{w_i}|\cdot) = \frac{\exp(e_{w_i} \cdot g(e_S, e_{w_{i-1}}, \ldots, e_{w_{i-N}}))}{\sum_w \exp(e_w \cdot g(e_S, e_{w_{i-1}}, \ldots, e_{w_{i-N}}))}
\]

(3.4)

where \( \sigma(\cdot) \) denotes sigmoid function.

Parameters \( \Theta \) and embeddings are estimated by making MLE estimation:

\[
[\Theta, e_D, \{e_P\}, \{e_S\}, \{e_w\}] = \arg\max_{\Theta', e_D', \{e_P'\}, \{e_S'\}, \{e_w'\}} \prod_D p(D|\Theta', e_D', \{e_P'\}, \{e_S'\}, \{e_w'\})
\]

(3.5)

### 3.2.4 Details of Implementation

Parameters \( \Theta \) and word embeddings are to be estimated from the training corpus. Meanwhile, we also estimate embeddings of documents, paragraphs and sentences
given words they include and the correspondent embeddings. MLE estimation is implemented as is the same with previous work. A similar strategy can be found in [58]. The estimated embeddings can be used as feature for downstream applications.

We employ three forms of operational functions.

(1) Skip-gram model [73]:

\[
  f(e_\eta|g(e_{parent(\eta)}, \{e_{\eta', \eta'\in Sibling(\eta)}\}, \Theta)) = \sigma(e_\eta \cdot e_{parent(\eta)}) \prod_{\eta'\in Sibling(\eta)} \sigma(e_\eta \cdot e_{\eta'})
\]

(3.6)

(2) CBOW like model [73] which first averages the embeddings of parent and siblings and dot products with current node embedding:

\[
  f(e_\eta|g(e_{parent(\eta)}, \{e_{\eta', \eta'\in Sibling(\eta)}\}, \Theta)) = \sigma(e_\eta \cdot g(e_{parent(\eta)}, \{e_{\eta', \eta'\in Sibling(\eta)}\}))
  
  g(e_{parent(\eta)}, \{e_{\eta', \eta'\in Sibling(\eta)}\}) = \frac{1}{1 + |Sibling(\eta)|} (e_{parent(\eta)} + \sum_{\eta'\in Sibling(\eta)} e_{\eta'})
\]

(3.7)

(3) Concatenation model which takes sequence order information by first concatenating embeddings of parent and siblings and then projects the concatenated vector sharing same dimensionality with current node embedding:

\[
  f(e_\eta|g(e_{parent(\eta)}, \{e_{\eta', \eta'\in Sibling(\eta)}\}, \Theta)) = \sigma(e_\eta \cdot g(e_{parent(\eta)}, \{e_{\eta', \eta'\in Sibling(\eta)}\}))
  
  g(e_{parent(\eta)}, \{e_{\eta', \eta'\in Sibling(\eta)}\}) = \text{tanh}(W \cdot [e_{S}, e_{w_{i-1}}, ..., e_{w_{i-N}}])
\]

(3.8)

where \([\cdot]\) denotes the concatenation of vectors and \(W\) denotes the \((1 + N) \times K\) dimensional convolutional matrix. For concatenation approach, we use a drop-out [41, 101] rate of 0.5. The initializations of embeddings for sentences, paragraphs and documents are conducted by averaging embeddings for tokens they contain using tf-idf, similar as in [44].

### 3.3 Experimental Results

Dimensionality of vectors are set to 300. All reported results are based on embeddings trained from the same Wikipedia2014 dataset. For each subset, Paragraph Vector and Joint Learning use the same \(f(\cdot)\) and \(g(\cdot)\) as the model at the top.
### Table 3.1: Results on Word Similarity Task

<table>
<thead>
<tr>
<th>Model</th>
<th>WS-353</th>
<th>RG</th>
<th>MC</th>
<th>SCWS</th>
<th>RW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip-Gram</td>
<td>68.7</td>
<td>78.1</td>
<td>71.5</td>
<td>58.1</td>
<td>37.2</td>
</tr>
<tr>
<td>Paragraph Vector</td>
<td>69.2</td>
<td>77.8</td>
<td>72.9</td>
<td>58.0</td>
<td>39.6</td>
</tr>
<tr>
<td>Joint Learning</td>
<td>71.2</td>
<td>78.6</td>
<td>73.8</td>
<td>57.9</td>
<td>41.7</td>
</tr>
<tr>
<td>CBOV</td>
<td>61.7</td>
<td>77.8</td>
<td>64.5</td>
<td>57.2</td>
<td>33.8</td>
</tr>
<tr>
<td>Paragraph Vector</td>
<td>62.4</td>
<td>79.1</td>
<td>65.8</td>
<td>56.9</td>
<td>34.2</td>
</tr>
<tr>
<td>Joint Learning</td>
<td>64.2</td>
<td>79.2</td>
<td>66.4</td>
<td>57.2</td>
<td>37.1</td>
</tr>
<tr>
<td>Concatenation</td>
<td>70.1</td>
<td>76.0</td>
<td>72.3</td>
<td>54.6</td>
<td>35.2</td>
</tr>
<tr>
<td>Paragraph Vector</td>
<td>70.0</td>
<td>77.1</td>
<td>72.5</td>
<td>57.2</td>
<td>37.9</td>
</tr>
<tr>
<td>Joint Learning</td>
<td>71.7</td>
<td>77.5</td>
<td>74.8</td>
<td>57.0</td>
<td>39.4</td>
</tr>
<tr>
<td>GloVe</td>
<td>68.6</td>
<td>77.5</td>
<td>77.2</td>
<td>52.7</td>
<td>39.2</td>
</tr>
</tbody>
</table>

**Word Similarity Evaluation**  Word embeddings are evaluated in terms of standard word similarity measures to see whether taking account of text hierarchy can improve those measures. We train our models using Wikipedia2014 dataset. We adopt a hierarchical softmax function for word prediction. The window size is set to 11. We keep the top 200,000 most frequent tokens\(^1\). We adopt a decreasing learning as applied in Word2Vec with initial learning rate is set to 0.05 and iterate three times over the corpus.

We employ standard ontology evaluation metrics include WS-353 [28], MC [78], RG [91], SCWS [44], and RW [65]. Each dataset is composed of pairs of words with gold-standard human annotations, indicating the similarity score between the pair of words. For example, “book, paper, 7.46” denotes the similarity score for word pair (book, paper) is 7.46. Standardly, we adopt cosine similarity. Spearman’s rank correlation coefficient is then obtained between this score and human judgement. Baselines include Skip-Gram, CBOV, Concatenation paragraph vector [58]. Experimental results are illustrated in Table 3.1. As can be seen, by considering the hierarchical structure of text units, better performance

---

\(^1\)Word embeddings and convolutional parameters are randomly initialized from [-0.1,0.1].
Table 3.2: Results on Word Analogy Task

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip-Gram</td>
<td>0.691</td>
</tr>
<tr>
<td>Paragraph-Vector</td>
<td>0.690</td>
</tr>
<tr>
<td>Joint Learning</td>
<td>0.714</td>
</tr>
<tr>
<td>CBOW</td>
<td>0.657</td>
</tr>
<tr>
<td>Paragraph-Vector</td>
<td>0.662</td>
</tr>
<tr>
<td>Joint Learning</td>
<td>0.678</td>
</tr>
<tr>
<td>Concatenation</td>
<td>0.702</td>
</tr>
<tr>
<td>Paragraph-Vector</td>
<td>0.706</td>
</tr>
<tr>
<td>Joint Learning</td>
<td>0.718</td>
</tr>
<tr>
<td>GloVe</td>
<td>0.716</td>
</tr>
</tbody>
</table>

has been achieved.

**Word Analogy Task** The models are trained on the same Wikipedia2014 corpus. Skip-Gram and CBOV are trained using Word2Vec.

We further test the proposed model on the word analogy task. Word analogy evaluation aims at answering questions like “a is to b as c is to what”. Question types include semantic ones like “Beijing is to China like London to what” (Capital) or syntactic ones like “dance to dancing as fly to what” (Tense). The dataset is introduced in [73] and contains 8,869 semantic questions and 10,675 syntactic questions. We follow the protocols described in [77, 47] that to answer questions “a is to b as c is to what”, we do the simple maths by computing $E_b - E_a + E_c$, where E denotes the embedding for current word, and find the word $d$ with the closest representation based on cosine similarity.

Performances regarding different models are illustrated in Table 3.2. Similar phenomena are observed on word similarity tasks where better performances are obtained when text structure is considered. The proposed model gives better performances than previous models for word embeddings since we consider both the local and global information.
3.4 Conclusion & Analysis

In this chapter, we present a hierarchical neural network model for word embeddings learning. Experiments verify the effectiveness of the learned word embeddings. The major aim of this work is to utilize text structure to improve word embeddings. Although we also generate embeddings for sentences, paragraphs and documents, we do not expect them to produce satisfying performances for a range of tasks without further improvements.

As stated, the learning of large text unit embeddings remains a problem. Due to data sparsity, it is not easy to learn representations for long text directly. Some work tries to learn the representation for sentences from the words they contain. However, as the compositionality for text is more complicated, tentative work shows limited improvements [58]. Then some work turns to use semantics. Vectors are used to represent the semantic relations of words, such as knowledge graphs and logical rules [114], to construct sentence vectors using both word vectors and semantic vectors. The problem is that relations are not as explicit as the co-occurrence of words. Also we found that in text, relations between components of different levels are rather different. We believe a simple vector representation is far from sufficient to model the different and complex relations between components of text. While to learn a unified representation of language mechanism is beyond the capability of natural language processing research now. In other words, a single vector is not enough to represent the information carried by sentences or paragraphs. More sophisticated methods are in need to represent meaning of long text. Our work in Chapter 6 further explores this point using vectors of entities to represent meaning of sentences.
Chapter 4

Empty Category Detection

4.1 Introduction

In discourse analysis, by nature we assume we have observed all the words when a piece of text is presented. But sometimes some words are omitted from the text for simplicity or other reasons. In this section, we focus on an important phenomena in languages, the empty category. The empty category (EC) is an important concept in linguistic theories. It is used to describe nominal words that do not have explicit phonological forms (they are also called “covert nouns”). This kind of grammatical phenomena is usually caused by the omission or dislocation of nouns or pronouns. Empty categories are the “hidden” parts of text and are essential for syntactic parsing [29, 125]. As a basic problem in NLP, the resolution of ECs also has a huge impact on a variety of downstream tasks, such as co-reference resolution [84, 52], long distance dependency relation analysis [70, 122]. Research also uncovers the important role of ECs in machine translation. Some recent work [18, 119] demonstrates the improvements they manage to obtain through EC detection in Chinese-English translation.

To resolve ECs, we need to decide 1) the position and type of the EC and 2) the content of the EC (to which element the EC is linked to if plausible). Existing research mainly focuses on the first problem which is referred to as EC detection [13, 125], and so is this work. As ECs are words or phrases inferable from their context, previous work mainly designs features mining the contexts of ECs and
then trains classification models or parsers using these features [123, 49, 29, 53]. One problem with these human-developed features are that they are not fully capable of representing the semantics and syntax of contexts. Besides, the feature engineering is also time-consuming and labour-intensive.

Recently neural network models have proven their superiority in capturing features using low dense vector compared with traditional manually designed features in dozens of NLP tasks [5, 19, 99, 20].

This chapter demonstrates the advantages of distributed representations and neural networks in predicting the locations and types of ECs. We formulate the EC detection as an annotation task, to assign predefined labels (EC types) to given contexts. Recently, [115] proposed a system taking advantages of the hidden representations of neural networks for image annotation which is to annotate images with a set of textual words. Following the work, we design a novel method for EC detection. We represent possible EC positions using the word embeddings of their contexts and then map them to a low dimension space for EC detection.

Experiments on Chinese Treebank show that the proposed model obtains significant improvements over the previous state-of-the-art methods based on strict evaluation metrics. We also identify the dependency relations between ECs and their heads, which is not reported in previous work. The dependency relations can help us with the resolution of ECs and benefit other tasks, such as full parsing and machine translation in practice.

4.2 Proposed Method

We represent each EC as a vector by concatenating the word embeddings of its contexts. As is shown in Fig. 1, we learn a map $MAP_A$ from the annotated data, to project the ECs’ feature vectors to a low dimension space $K$. Meanwhile, we also obtain the distributed representations of EC types in the same low dimension space $K$. In the testing phase, for each possible EC position, we use $MAP_A$ to project its context feature to the same space and further compare it with the representations of EC types for EC detection.

Distributed representations are good at capturing the semantics and syntax of
4.2. PROPOSED METHOD

Figure 4.1: System Architecture of EC Detection

contexts. For example, with word embeddings we are able to tell that “吃/eat” and “喝/drink” have a closer relationship than “吃/eat” and “走/walk” or “喝/drink” and “走/walk”. Thus the knowledge we learn from: “EC(你/You)-吃/have-EC(晚饭/supper)-了/past tense marker-么/question marker” could help us to detect ECs in sentences such as “EC(你/You)-饮料/beverage-喝/drink-了/past tense marker-么/question marker”, which are similar, though different from the original sentence.

Below is a list of EC types contained in the Chinese Treebank, which are also the types of EC we are to identity in this work.

- pro: little pro, refer to dropped pronouns.
- PRO: big PRO, refer to shared elements in control structures or elements that have generic references.
- OP: null operator, refer to empty relative pronouns.
- T: trace left by A’-movement, e.g., topicalization, relativization.
- RNR: used in right nodes rising.
- *: trace left by passivization, raising.
- Others: other ECs.
According to the reason that one EC is caused, we are able to assign it one of the above categories.

We can formulate EC detection as a combination of a two-class classification problem (is there an EC or not) and a seven-class classification problem (what type the EC is if there is one) following the two-pass method. For one-pass method, EC detection can be formulated as an eight-class (seven EC types listed above plus a dummy “No” type) classification problem. Previous research shows there is no significant differences between their performances [123]. Here we adopt the one-pass method for simplicity.

### 4.2.1 System Overview

The proposed system consists of two maps.

$MAP_A$ is from the feature vector of an EC position to a low dimensional space.

$$MAP_A : R^n \rightarrow R^k, k \ll n$$

$$f_A(X) \rightarrow W_A X$$

$MAP_A$ is a linear transformation, and $W_A$ is a $k \times n$ matrix.

The other one is from labels to the same low dimensional space.

$$MAP_B : \{Label_1, Label_2, \ldots\} \in R \rightarrow R^k$$

$$f_B(Label_i) \rightarrow W^i_B$$

$MAP_B$ is also a linear transformation. $W^i_B$ is a $k$ dimensional vector and it is also the distributed representation of $Label_i$ in the low dimensional space.

The two maps are learned from the training data simultaneously. In the testing phase, for any possible EC position to be classified, we extract the corresponding feature vector $X$, and then map it to the low dimensional space using $f_A(X) = W_A X$. Then we have $g_i(X)$ for each $Label_i$ as follows:

$$g_i(X) = (f_A(X))^T W^i_B$$

For each possible label $Label_i$, $g_i(X)$ is the score that the example having a $Label_i$ and the label predicted for the example is the $i$ that maximizes $g_i(X)$. 
4.2. PROPOSED METHOD

Following the method of [115], we try to minimize a weighted pairwise loss, learned using stochastic gradient descent:

$$\sum_X \sum_{i \neq c} L(\text{rank}_c(X)) \max(0, (g_i(X) - g_c(X)))$$

(4.4)

Here $c$ is the correct label for example $X$, and $\text{rank}_c(X)$ is the rank of $Label$ $c$ among all possible labels for $X$. $L$ is a function which reflects our attitude towards errors. A constant function $L = C$ implies we aim to optimize the full ranking list. Here we adopt $L(\alpha) = \sum_{i=1}^{\alpha} 1/i$, which aims to optimize the top 1 in the ranking list, as stated in [105]. The learning rate and some other parameters of the stochastic gradient descent algorithm are to be optimized using the development set.

An alternative method is to train a neural network model for multi-class classification directly. It is plausible when the number of classes is not large. One of the advantages of representing ECs and labels in a hidden space is that EC detection usually serves as an intermediate task. Usually we want to know more about the ECs such as their roles and explicit content. Representing labels and ECs as dense vectors will greatly benefit other work such as EC resolution or full parsing. Besides, such a joint embedding framework can scale up to the large set of labels as is shown in the image annotation task [115], which makes the identification of dependency types of ECs (which is a large set) possible.

4.2.2 Context Features Construction

Defining Locations

In a piece of text, possible EC positions can be described with references to tokens, e.g., before the $n^{th}$ token [125]. One problem with such methods is that if there are more than one ECs preceding the $n^{th}$ token, they will occupy the same position and can not be distinguished. One solution is to decide the number of ECs for each position, which complicates the problem. But if we do nothing, some ECs will be ignored.

A compromised solution is to describe positions using parse trees [123]. Adjacent ECs before a certain token usually have different head words, which means
they are attached to different nodes (head words) in a parse tree. Therefore, it is possible to define positions using “head word, following word” pairs. The problem of EC detection is formulated as a classification problem: for each “head word, following word” pair, what is the type of the EC? An example is shown in figure 4.2, in which there are 2 possible EC positions, (吃，了) and (吃，。)\(^1\).

Besides, we keep punctuations in the parse tree so that we can describe all the possible positions using the “head node, following word” pairs, as no elements will appear after a full stop in a sentence.

**Feature Extraction**

The feature vector is constructed by concatenating the word embeddings of context words that are expected to contribute to the detection of ECs.

1. The head word (except the dummy root node). Suppose words are represented using \(d\) dimension vectors, we need \(d\) elements to represent this

\(^1\)Note that there are still problems with the tree-based method. As is shown in Fig. 4.3, the pro and T are attached to the same head word (告别) and share the same following word (德国). But such cases are quite rare in the dataset we use, here we still adopt the tree-based method.
4.2. PROPOSED METHOD

feature. The distributed representations of the head word would be placed at the corresponding positions.

2. The following word in the text. This feature is extracted using the same method with head words.

3. “Nephews”, the sons of the following word. We choose the leftmost two.

4. Words in dependency paths. ECs usually have long distance dependencies with words which cannot be fully captured by the above categories. We need a new feature to describe such long distance semantic relations: Dependency Paths. From the training data, we collect all the paths from root nodes to ECs (ECs excluded) together with dependency types. Below we give an example to illustrate the extraction of this kind of features using a complex sentence with a multi-layer hierarchical dependency tree as in Fig. 4.3. If we have $m$ kinds of such paths with different path types or dependency types, we need $md$ elements to represent this kind of features. The distributed representations of the words would be placed at the corresponding positions in the feature vector and the remaining are set to 0.

Previous work usually involves lots of syntactic and semantic features. In the work of [123], 6 kinds of features are used, including those derived from constituency parse trees, dependency parse trees, semantic roles and others. Here
we use only the dependency parse trees for the feature extraction. The words in dependency paths we use have proven their potential in representing the meanings of text in frame identification [38].

Take the OP in the sentence shown in Fig. 4.3 for example. For the OP, its head word is “的”, its following word is “告别” and its nephews are “NULL” and “NULL” (ECs are invisible).

The dependency path from root to OP is:
\[
\text{Root } \xrightarrow{\text{ROOT}} \text{举行/hold } \xrightarrow{\text{COMP}} \text{仪式/ceremony } \xrightarrow{\text{RELC}} \text{的/DE } \xrightarrow{\text{COMP}} \text{OP}
\]

For such a path, we have the following sub-paths:
\[
\begin{align*}
\text{Root } & \xrightarrow{\text{ROOT}} . \xrightarrow{\text{COMP}} . \xrightarrow{\text{RELC}} X \\
\text{Root } & \xrightarrow{\text{ROOT}} . \xrightarrow{\text{COMP}} X \\
\text{Root } & \xrightarrow{\text{ROOT}} X
\end{align*}
\]

For the position of the OP in the given example, the words with corresponding dependency paths are “的”, “仪式” and “举行”. Similarly, we collects all the paths from other ECs in the training examples to build the feature template.

In the testing phase, for each possible EC position, we place the distributed representations of the right words at the corresponding positions of its feature vector.

### 4.3 Experiments on CTB

#### 4.3.1 Data

The proposed method can be applied to various kinds of languages as long as annotated corpus are available. In our experiments, we use a subset of Chinese Treebank V7.0.

We split the data set into three parts, training, development and test data. Following the previous research, we use File 1-40 and 901-931 as the test data, File 41-80 as the development data. The training data includes File \{81-325, 400-454, 500-554, 590-596, 6000-885, 900\}. The development data is used to tune parameters and the final results are reported on the test data. CTB trees are transferred to dependency trees for feature extraction with ECs preserved [121].
4.3. EXPERIMENTS ON CTB

The distributed word representation we use is learned using the word2vec toolkit [76]. We train the model on a large Chinese news corpora provided by Sogou\(^2\), which contains about 1 billion words after necessary preprocessing. The text is segmented into words using ICTCLAS\(^3\).

4.3.2 Experiment Settings

**Initialization** \(W_A\) is initialized according to \(\text{uniform}[\frac{-24}{d_{\text{in}}+d_{\text{hidden}}}, \frac{-24}{d_{\text{in}}+d_{\text{hidden}}}].\) And \(W_B\) is initialized using \(\text{uniform}[\frac{-24}{d_{\text{hidden}}+d_{\text{out}}}, \frac{24}{d_{\text{hidden}}+d_{\text{out}}}].\)

Here \(d_{\text{in}}, d_{\text{hidden}}\) and \(d_{\text{out}}\) are the dimensions of the input layer, the hidden space and the label space.

**Parameter Tuning** To optimize the parameters, firstly, we set the dimension of word vectors to be 80, the dimension of the hidden space to be 50. We search for the suitable learning rate in \(\{10^{-1}, 10^{-2}, 10^{-4}\}\). Then we deal with the dimension of word vectors \(\{80, 100, 200\}\). Finally we tune the dimension of hidden space

\(^2\)http://www.sogou.com/labs/dl/cs.html
\(^3\)The word segment standards used by CTB and ICTCLAS are roughly the same with minor differences.
in \{50, 200, 500\} against the F-1 scores. Those underlined figures are the value of the parameters after optimization. We use the stochastic gradient descent algorithm to optimize the model. The details can be checked here [115]. The maximum iteration number we used is 10K. In the following experiments, we set the parameters to be learning rate=\(10^{-1}\), word vector dimension=80 and hidden layer dimension=500.

From the experiments for parameter tuning, we find that for the word embeddings in the proposed model, low dimension vectors are better than high dimension
ones for low dimension vectors are better in sharing meanings. For the hidden space which represents inputs as uninterpreted vectors, high dimension vectors are better than low dimension vectors. The learning rates also have an impact on the performance. If the learning rate is too small, we need more iterations to achieve convergence. If we stop iterations too early, we will suffer under-fitting.

4.3.3 Results

Metrics and Evaluation

Previous work reports results based on different evaluation metrics. Some work uses linear positions to describe ECs. ECs are judged on a “whether there is an EC of type A before a certain token in the text” basis [13]. Collapsing ECs before the same token to one, [13] has 1352 ECs in the test data. [123] has stated that some ECs that share adjacent positions have different heads in the parse tree. They judge ECs on a “whether there is an EC of type A with a certain head word and a certain following token in the text” basis. Using this kind of metric, they get 1765 ECs.

Here we use the same evaluation metric with [123]. Note that we still cannot describe all the 1838 ECs in the corpora, for on some occasions ECs preceding the same token share the same head word. We also omit some ECs which cause cycles in dependency trees as described in the previous sections. We have 1748 ECs, 95% of all the ECs in the test data, very close to 1765 used by [123]. The total number of ECs has an impact on the recall. In Table 4.4, we include results based on each method's own EC count (1748, 1765, 1352 for Ours, Xue's and Cai's respectively) and the real total EC count 1838 (figures in brackets).

[125] report an experiment result based on a classification model in a unified parsing frame. We do not include it for it uses different and relatively loose evaluation metrics. The distributions of ECs in the test data are shown in Table 4.3.

The results are shown in Table 4.4. We present the results for each kind of EC and compare our results with two previous state-of-the-art methods[13, 123].

The proposed method yields the newest state-of-the-art performances on CTB
Table 4.3: EC Distribution in the Test Data

<table>
<thead>
<tr>
<th>Type</th>
<th>PRO</th>
<th>pro</th>
<th>T</th>
<th>OP</th>
<th>RNR</th>
<th>*</th>
<th>Others</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>297</td>
<td>298</td>
<td>575</td>
<td>527</td>
<td>32</td>
<td>19</td>
<td>0</td>
<td>1748</td>
</tr>
<tr>
<td>Xue</td>
<td>305</td>
<td>298</td>
<td>584</td>
<td>527</td>
<td>32</td>
<td>19</td>
<td>0</td>
<td>1765</td>
</tr>
<tr>
<td>Cai</td>
<td>299</td>
<td>290</td>
<td>578</td>
<td>134</td>
<td>32</td>
<td>19</td>
<td>0</td>
<td>1352</td>
</tr>
</tbody>
</table>

Table 4.4: Performance on the CTB Test Data

<table>
<thead>
<tr>
<th>Class</th>
<th>Correct</th>
<th>p</th>
<th>r</th>
<th>F-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRO</td>
<td>162</td>
<td>.479</td>
<td>.545</td>
<td>.510</td>
</tr>
<tr>
<td>pro</td>
<td>161</td>
<td>.564</td>
<td>.540</td>
<td>.552</td>
</tr>
<tr>
<td>OP</td>
<td>409</td>
<td>.707</td>
<td>.776</td>
<td>.740</td>
</tr>
<tr>
<td>T</td>
<td>506</td>
<td>.939</td>
<td>.88</td>
<td>.908</td>
</tr>
<tr>
<td>RNR</td>
<td>23</td>
<td>.767</td>
<td>.719</td>
<td>.742</td>
</tr>
<tr>
<td>*</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Overall (Xue)</td>
<td>1261</td>
<td>.712</td>
<td>.721 (.686)</td>
<td>.717 (.699)</td>
</tr>
<tr>
<td>(Caï)</td>
<td>903</td>
<td>.653</td>
<td>.512 (.491)</td>
<td>.574 (.561)</td>
</tr>
<tr>
<td></td>
<td>737</td>
<td>.660</td>
<td>.545 (.401)</td>
<td>.586 (.499)</td>
</tr>
</tbody>
</table>
4.3. EXPERIMENTS ON CTB

as far as we know. We also identify the dependency types between ECs and their heads. Some ECs, such as pro and PRO, are latent subjects of sentences. They usually serve as SBJ with very few exceptions. The others may play various roles. There are 31 possible (EC, Dep) pairs. Using the same model, the overall result is $p = 0.701, r = 0.703, f = 0.702$.

Analysis

We compare the effectiveness of different features by ablating each kind of features described in the previous section. As Table 4.5 shows, the most important kind is the dependency paths, which cause a huge drop in performance if ablated. Dependency paths encode words and path pattern information, which is proved essential for the detection of ECs. Besides, headwords are also useful. While for the others, we cannot easily make the conclusion that they are of little usage in the identification of ECs. They are not fully explored in the proposed model, but may be vital for EC detection.

Worth to mention is that of the several kinds of ECs, the proposed method shows the best performance on ECs of type T, which represents ECs that are the trace of A’-movement, which moves a word to a position where no fixed grammatical function is assigned. An example is presented in Fig. 4.5.

“A” is moved to the head of the sentence as the topic (topicalization) and left a trace which is the EC. To detect this EC, we need information about the action “喜欢/like”, the link verb “看起来/seem” and the arguments “A” and “B”. ECs of type $T$ are very common in Chinese, since Chinese is a topic-prominent language. Using distributed representations, it is easy to encode the context information in our feature vectors for EC detection.

We also report satisfying results and significant improvements for the other
types except ‘*’ (trace of A-movement), which make up about 1% of all the ECs in the test data. This is mainly because that there are too few ‘*’ examples in the training data. We need to further improve our models to detect such ECs.

### 4.4 Discussion

The proposed method is capable of handling large set of labels. Hence it is possible to detect EC types and dependency types simultaneously. Besides, some other NLP tasks can also be formulated as annotation tasks, and therefore can be resolved using the same scheme, such as the frame identification for verbs [38].

This work together with some previous work that uses classification methods [13, 123, 121], regards ECs in a sentence as independent to each other and even independent to words that do not appear in the feature vectors. Such an assumption makes it easier to design models and features but does not reflect the grammatical constraints of languages. For example, simple sentences in Chinese contain one and only one subject, whether it is an EC or not. If it is decided there is an EC as a subject in a certain place, there should be no more ECs as subjects in the same sentence. But such an important property is not reflected in these classification models. Methods that adopt parsing techniques take the whole parse tree as input and output a parse tree with EC anchored. So we can view the sentence as a whole and deal with ECs with regards to all the words in the sentence. [45] also take the grammar constraints into consideration by formulating EC detection as an integer linear problem (ILP) problem. But they usually yield poor performances compared with classification methods partly because the methods they use can not fully explore the syntactic and semantic features.
4.5 Related Work

Existing methods for EC detection mainly explore syntactic and semantic features using classification models or parsing techniques.

[49] proposes a simple pattern based algorithm to recover ECs, both the positions and their antecedents in phrase structure trees. [29] presents a two stage parser that uses syntactic features to recover Penn Treebank style syntactic analysis, including the ECs. The first stage, sentences are parsed as usual without ECs, and in the second stage, ECs are detected using a learned model with rich text features in the tree structures. [53] reports a tree kernel-based model which takes as input parse trees for EC detection. They also deal with EC resolution, to link ECs to text pieces if possible. They report their results on Chinese Treebank. [125] restores ECs from parse trees using a Maximum Entropy model. [45] proposes a cross-lingual ILP-based model for zero anaphora detection. [13] reports a classification model for EC detection. Their method is based on “is there an EC before a certain token”.

Recently [123] further develop the method of [125] and explore rich syntactic and semantic features, including paths in parse trees and semantic roles, to train an ME classification model for EC detection and yield the best performance reported using a strict evaluation metric on Chinese Treebank as far as we know.

As we have stated, the traditional features used by above methods are not good at capturing the meanings of contexts. Currently the distributed representations together with deep neural networks have proven their ability not only in representing meaning of words, inferring words from the context, but also in representing structures of text [99]. Deep neural networks are capable of learning features from corpus, therefore saves the labour of feature engineering and have proven their ability in lots of NLP task [20, 5].

The most relevant work to this chapter are that of [115] and that of [38]. [115] proposes a deep neural network scheme exploring the hidden space for image annotation. They map both the images and labels to the same hidden space and annotate new images according to their representations in the hidden space. [38] extends the scheme to frame identification, for which they obtain satisfying
results. This work further uses it for empty category detection with novel features.

Compared with previous research, the proposed model simplifies the feature engineering greatly and produces distributed representations for both ECs and EC types which will benefit other tasks.

4.6 Conclusion

In this chapter, we propose a new empty category detection method using distributed word representations. Using the word embeddings of the contexts of ECs as features enables us to employ rich information in the context without much feature engineering. Experiments on CTB have verified the advantages of the proposed method. We successfully beat the existing state-of-the-art methods based on a strict evaluation metric. The proposed method can be further applied to other languages such as Japanese.
Chapter 5

Entity-Based Coherent Summarization

5.1 Introduction

Using entities discovered in discourse, including the covert ones, we are able to build an entity-centric view of discourse. Entities are regarded as the core of discourse and discourse is elaborated around entities. We use the entity-centric representations for different tasks. In this chapter, we present a summarization system using entities.

Automatic summarization is extremely useful in this age of information overload. It provides readers with easier access to information without the labour of reading the source text. According to the number of documents dealt with, summarization falls into two categories: single-document summarization and multi-document summarization. While they both aim to represent the source text using a shortened passage, the latter deals with a set of documents sharing the same topic. Based on the method adopted, existing approaches to summarization can be divided into two kinds: the abstraction-based methods and the extraction-based methods. The difference lies in the sentences they use to generate summaries: the former selects sentences (clauses, or other text units, hereafter we refer to all of them as sentences.) from source documents and the latter generates new sentences. Most existing summarization systems are extraction-based because
abstraction-based methods require the use of natural language generation technology, which is still a growing field. This work, without exception, also employs extraction-based methods.

Currently the extraction-based methods face some major challenges. One is informativeness, which means we need to maintain the important information of source documents in summaries. This is the focus of almost all research on summarization. Another challenge is presentation, namely that the extracted text should be well presented, i.e., it should contain little redundancy and be coherent so as to be readily understandable. Previous work has addressed the problem of redundancy, and some successful solutions like Maximum Marginal Relevance (MMR) [14] have been proposed and widely adopted, but very few try to deal with coherence. Therefore, the generated summaries generally suffer as regards readability and are very difficult to use for practical applications. In the report of the TAC 2011 summarization task [82], it is stated that “in general, automatic summaries are better than baselines\textsuperscript{1}, except Readability.” Such a statement suggests, as for summarization, coherence should be treated with the same as salience and redundancy.

Existing work addresses coherence in summarization from different aspects. One kind of method employs reordering after selecting sentences, and the drawback is evident: coherence is considered after sentence selection. Another kind of widely adopted method takes discourse relations into consideration when selecting sentences, as discourse relations are believed to be essential for maintaining textual coherence. [42] formulated single document summarization as to extract a sub tree from the complete discourse tree and thus preserve the relations between extracted document units to form a readable text. [113] extended it to multi-document summarization by regarding a document set as one document and developed a model which combined discourse parsing and summarization together. [16] proposed a graph-based model to bypass the tree constraints. They employed rich textual features to build a discourse relation graph for source documents with the aim of representing the relations between sentences (both inter

\textsuperscript{1}The baseline they used is the lead paragraph method and summaries are evaluated by human and ROUGE (Recall-Oriented Understudy for Gisting Evaluation [62]).
and intra-document relations). [16] reported ROUGE scores lower than some baselines. This is because that, they claim, ROUGE is salience-focused and fails to notice the improvement in coherence. In a further human evaluation, they reported improvements in readability.

These discourse-based methods without exception have discourse analysis as a prerequisite. As we all know, discourse analysis is still under development thus preventing the expected improvement. Furthermore, languages other than English do not enjoy plenty of ready-to-use discourse analysis tools. This also limits the usage of these discourse-based methods.

Is it possible to consider coherence in summarization without discourse analysis? Before answering this question, we need to find out what is the key to coherence in text. According to the centering theory [36, 107], the coherence of text is to a large extent maintained by entities and the relations between them. This indicates that discourse analysis is not a must to preserve coherence; we can directly take advantages of entities and their relations to generate coherence text.

Based on this point, we design a novel graph-based model for multi-document summarization that eliminates the effort of conducting discourse relation analysis (inter or intra document) and generates informative and readable summaries. We formulate the document set as a graph whose nodes correspond to sentences. These nodes are connected with each other according to the entities they contain. Each path in the graph represents a piece of text and is evaluated using a novel scoring function that considers informativeness and coherence. To extract a summary is to find a path in the graph with the highest score. This is a weighted longest path problem. We further present a variant of the proposed model based on local coherence and explore decoding algorithms for both of them.

Experiments are conducted on the Document Understanding Conference (DUC) 2004 multi-document summarization task data set. As ROUGE cannot fully capture our improvement in coherence which is one of the key contributions of this work, we also conduct a human evaluation. Results show that we obtain summaries comparative with state-of-the-art systems in terms of ROUGE metrics and get improvements in readability in human evaluations.

This work provides a method of generating high quality summaries without
the effort of discourse analysis. The proposed method can be easily extended to other languages. It also provides inspiration as regards to other tasks that require computers to generate coherent text. The rest of the chapter is organized as follows: Section 5.2 presents the centering theory and a coherence model based on entities. Section 5.3 presents our model. Section 5.4 describes the experiments and results. Section 5.5 presents some previous work and Section 5.6 concludes this chapter.

5.2 Centering Theory and Coherence Modelling

The centering theory [36] as a popular theory on discourse analysis, serves as the basis of some coherence evaluation methods [3, 12] and enables us to measure the coherence score of any given text without discourse parsing and solely based on the reappearance of entities. Entities here refer to noun/pronoun word/phrases.

According to the centering theory, we have the following assumptions:

1. Text that contains successive mentions of the same entities would be more coherent.

2. The main entities that are focused on tend to play an important grammatical role, such as the subject or object of the sentences.

Therefore the key to the coherence of a text lies in what entities it contains and how their roles change. The coherence of a generated text can be evaluated accordingly.

[3] presented such a model. The key is to represent text as an entity grid. Assume text $T$ contains $n$ sentences $\{S_1, S_2, ..., S_n\}$ and $m$ entities. $r_i^k$ represents the grammatical role of Entity $e_k$ in Sentence $S_i$. Four kinds of roles are used, i.e., “subj”, “obj”, “others” and “absent”. “Others” indicates that the entity is present, but is neither the subject nor the object. Then the grammatical roles of

\[\text{In some previous work on summarization [104, 42], concepts are used to measure informativeness. Concepts can be used to refer any non-functional words, including adjectives, adverbs. All the entities can be regarded as concepts, but some concept words (non-nominal words) are not entities. Entity is a subset of Concept.}\]
5.3. MODELLING SUMMARIZATION

e_k in text T can be expressed as a sequence: \( \{r_1^k, r_2^k, \ldots, r_n^k\} \). For each entity in T, such a chain showing how the entity’s grammatical roles change in T is extracted. Thus text T can be represented as an \( n \times m \) matrix \( M(T) \) where \( n \) is the number of sentences and \( m \) is the number of entities in T, and \( M(T)_{ij} \) corresponds to the grammatical roles of Entity j in Sentence i. \( M(T) \) is referred to as the Entity Grid of T [3].

To calculate the coherence score of T, [3] used \( M(T) \) as a feature vector. They calculated the transition probability for \( |\{(s(subj), o(bj), x(others), \neg(absent))\}| = 16 \) transition patterns from \( M(T) \) without distinguishing between entities, to form a vector \( f(T) \) for T, and a weight vector \( w \) was then learnt from training data so that \( w \ast f(T) \) can be used as the coherence score for T.

This kind of methods has been adopted in many studies [27, 3, 12]. In particular, [26] extends entity grids to model semantic relations between entities, which provides a possible further improvement for our models.

5.3 Modelling Summarization

The above model can only be used to measure coherence but summarization is much complex as it involves not only coherence but also informativeness and redundancy. We design a much more sophisticated models leveraging entities. Two models are presented below. Both of them are based on entities and consider coherence as well as informativeness. The first one is based on global coherence and the second one local coherence. The global coherence consider the full sequence when evaluating coherence and the local coherence is calculated based on relations between adjacent sentences. Intuitively, global coherence is better than local coherence, but considering the full sequence increases the time complexity. The model based on local coherence, on the other hand, reduces the time complexity and enables us to obtain an exact solution efficiently.

5.3.1 Problem Setup

Assume we have \( K \) documents with \( n \) sentences in total. Note that we are dealing with multi-document summarization, and we do not distinguish between inter-
document and intra-document relations. We construct a graph with \( n \) nodes, each of which corresponds to one sentence. Weighted directed edges are used to connect these nodes together. To each node, we assign a cost score, which is the number of words the corresponding sentence contains. To each path in the directed graph, we assign a gain score. The gain score is a comprehensive evaluation of the informativeness and coherence of the sequence of sentences represented by the path. The problem of extracting a good summary becomes the problem of extracting the best path. Note that it is an asymmetric graph. Gain scores for \( A \rightarrow B \rightarrow C \) and \( C \rightarrow B \rightarrow A \) are different. The direction determines the positions of corresponding sentences in the generated text.

One more thing to consider is the redundancy. Instead of formulating redundancy explicitly, we remove edges connecting similar sentences to turn the complete graph into an incomplete graph. This ensures that similar sentences do not occupy adjacent positions in the generated summaries and thus reduce redundancy. The similarities of sentence pairs are based on word overlaps, and we keep \( d\% \) of all the edges.

Note that for temporal text removing edges can also help us maintain the temporal relations between sentences, though we do not explore this point here.

### 5.3.2 Summarization Considering Global Coherence

To extract a summary is to find such a sequence of sentences \( Seq \) that maximizes \( \text{Score}(Seq) \).

\[
\text{Score}(Seq) = \sum_{k=1}^{m} a_k F_k
\]

\[
F_k = \prod_{i} p_{e_k}(r^k_i, r^k_{i+1}), S_i, S_{i+1} \in Seq
\]

s.t. \( \sum_{S_i \in Seq} \text{length}(S_i) \leq \text{threshold} \) (5.1)

\( a_k \) is the weight of Entity \( e_k \). \( r^k_i \) is the state of Entity \( e_k \) in Sentence \( S_i \). Here we use four states: “s”, “o”, “x”, “-”, which represent “subj”, “obj”, “present” and “absent” respectively. It is also possible to use more or fewer states.
5.3. MODELLING SUMMARIZATION

\( p_{e_k}(***) \) is the transition probability between two states for \( e_k \). For each document set, the transition probabilities for each entity is estimated using \( p_{e_k}(ab) = \frac{\#e_k(a)e_k(b)}{n-K} \). \#e_k(a)e_k(b) marks the times that Entity \( e_k \) presents as grammatical role \( a \) in the preceding sentence and as grammatical role \( b \) in the following one. \( n - K \) denotes the total number of adjacent sentence pairs in a document set with \( K \) documents and \( n \) sentences. \( F_k \) is the coherence score contributed by \( e_k \) in the extracted sequence \( Seq \). \( F_k \) is based on the transitions of \( e_k \) between adjacent sentences in \( Seq \). We use \( Score(Seq) \) which considers salience, coherence and redundancy as an index as to how suitable the extracted sentence sequence \( Seq \) is as a summary. This model is a weighted longest path problem with a fixed length.

This is an NP-hard problem. Due to the time cost, we adopt the simple randomized algorithm as shown in Algorithm 1 to obtain an approximated solution. Other decoding algorithms like greedy algorithms can also be employed.

**Algorithm 1** A randomized algorithm for the weighted longest path problem

Initialization:
Set \( U \) ← all the sentences in the current doc set
Set \( S \) ← EmptySet
Queue \( Q \) ← EmptySet
repeat
  randomly select sentence \( s \in U \& s \notin Q \);
  if \( \text{length}(s) + \sum_i \text{length}(s_i) \leq \text{threshold}, s_i \in Q \) then
    push \( s \) to the rear of \( Q \)
  else
    push \( Q \) into \( S \), Queue \( Q \) ← EmptySet
  end if
until 10K times
return \( \text{argmax}_Q F(T), Q \in S \)

But none of them are capable of obtaining an exact solution. Below we present another model considering local coherence which makes it possible to obtain an exact solution.
5.3.3 Summarization Considering Local Coherence

The above model considers global coherence which is calculated according to the whole text. The model presented below is directly based on local coherence and enables us to obtain an exact solution. We want to maximize $\text{Score}(\text{Seq})$:

$$
\text{Score}(\text{Seq}) = \sum_{S_i \in \text{Seq}} (\alpha \sum_{e_k \in S_i} a_k + (1 - \alpha) \text{gain}_{i,(i+1)})
$$

s.t. $\sum_{S_i \in \text{Seq}} \text{length}(S_i) \leq \text{threshold}$ \hspace{1cm} (5.2)

This formulation contains two parts. $\sum_{e_k \in S_i} a_k$ implies the weight of Sentence $S_i$, which is the sum of its entities’ weights. $\text{gain}_{i,(i+1)}$ is the gain score for $\text{Edge}(S_i, S_{i+1})$. $\alpha$ manipulates the impacts of the two parts.

$$
\text{gain}(S_i, S_{i+1}) = \sum_{e_k \in S_i \cup S_{i+1}} p_{e_k} (r^k_i r^k_{i+1})
$$

As is stated, $r^k_i$ is the state of Entity $e_k$ in Sentence $S_i$.

For the convenience of decoding, we turn the above model to an integer linear programming (ILP) problem. We add two dummy nodes, called Start and End Node. All paths start from Start and end with End. The costs of both Start and End are 0. The gains of edges connected with Start or End are 0. Note that although here we present a full connected graph for simplicity, in practise we delete several edges to reduce redundancy. Following such a setting, an arbitrary path in the old graph (the one without dummy Start and End nodes) can be represented as a path from Start to End. We write the Start node as Node 0 and the End node as Node $t$. Then we formulate the problem of the weighted longest
5.4. EXPERIMENTS & ANALYSIS

path as follows:

\[
\text{maximize} \alpha \sum_i \left( \sum_{e_k \in S_i} a_k \right) x_i + (1 - \alpha) \sum_{i,j} \text{gain}_{i,j} y_{ij}
\]

subject to

\[
\begin{align*}
1) \sum_i \text{cost}_i x_i & \leq \text{threshold} \\
2) \sum_i y_{0i} & = 1 \\
3) \sum_i y_{it} & = 1 \\
4) \sum_i y_{ij} + y_{0j} - (\sum_i y_{ji} + y_{jt}) & = 0, \forall j \\
5) \sum_i y_{ij} + y_{0j} - x_j & = 0, \forall j \\
6) x_i & \in \{0, 1\}, \forall i \\
7) y_{ij} & \in \{0, 1\}, \forall i, j
\end{align*}
\]

Equations 2 and 3 are used to ensure we have only one start and one end node. Equation 4 ensures that the in degree equals the out degree for all nodes. Equation 5 ensures that the in degree is either 0 or 1 and equals \(x_a\) for all nodes. \(x_i = 1\) indicates that \(S_i\) is selected for the summary. \(x_i = 0\) means \(S_i\) is not contained in the summary. \(y_{ij} = 1\) means \(S_i\) and \(S_j\) are selected and placed as adjacent sentences in the summary. \(\text{cost}_i\) is the number of words in \(S_i\) (length of \(S_i\)).

We resolve this ILP problem using the dual simplex method provided by IBM CPLEX optimizer\(^3\) which is a powerful optimization software package. CPLEX provides both a primal simplex method and a dual simplex method for ILP problems. Here we adopt the latter.

5.4 Experiments & Analysis

5.4.1 Experiment

Experiments are conducted on the data set of the DUC2004 Summarization Task, which is a multi-document summarization task. 50 document clusters, each of which consists of 10 documents, are given. One summary is to be generated for

\(^3\)http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/
each cluster. The target length is up to 100 words. Weights of entities are learnt by logistic regression as is adopted by [104] \(^4\). For entities that are not contained in DUC2003, we assign tf-based weights to them as [3] did.

For the evaluation we firstly use the generally acknowledged metric for summarization: ROUGE metric. It essentially calculates n-gram overlaps between automatically generated summaries and human written (the gold standard) summaries. A high level of overlap indicates a high level of shared information between the two summaries. Among others, we focus on ROUGE-1 in the discussion of the result, because ROUGE-1 has proved to have a strong correlation with human annotation [62].

Some necessary preprocessing includes stemming, removing stop-words and simple simplification. In previous work, there is usually no co-reference resolution and different words are regarded as different entities. For example, [3] regard “Microsoft”, “Microsoft Company” and “the company”\(^5\)” as different entities. Here we use Stanford CoreNLP toolkit [68] to deal with the co-reference problem. The Stanford CoreNLP toolkit contains a ready-to-use entity identification tool and a co-reference resolution tool. The NER toolkit reports an F-score of about 86\% on the CoNLL 2003 NER Shared Task data set and the co-reference tool reports an average F-score of about 60\% on the CoNLL 2001 Shared Tasks data set. The co-reference resolution is not a must, though preferred if reliable tools are available.

After the co-reference resolution, different forms of the same entities are replaced by their unified forms. For each document set, we need to estimate the transition probabilities for each entity according to the documents contained in the cluster as stated above. We have \( p_{e_k}(ab) = \frac{\#e_k(a)e_k(b)}{n-K} \). \( \#e_k(a)e_k(b) \) marks the times that Entity \( e_k \) presents as grammatical role \( a \) in the preceding sentence and as grammatical role \( b \) in the following one. \( n - K \) denotes the total number of adjacent sentence pairs in a document set with \( K \) documents and \( n \) sentences. As stated above, we use a logistic regression model to learn weights for entities

\(^4\)This method was first proposed by [126] and then improved by [104]. Here we follow the same steps with [104].

\(^5\)Here “the company” refers to Microsoft judged from the context.
from the DUC2003 data, as in the work of [104].

Parameters are tuned using the DUC2003 dataset. \( d \) is the threshold of redundancy. We keep \( d \) percent of all edges and \( d \) varies from 10 to 100 with an interval of 10. We tune the parameter using the randomized algorithm and evaluate the results using ROUGE-1 Recall. In the following experiments, we set \( d = 80 \), which means we keep 80% of the sentences.

As for the model presented in Section 5.3.3, we need to tune \( \alpha \). Using the same data, we try \( \alpha \) from 0 to 1 with an interval of 0.1 and eventually choose \( \alpha = 0.4 \).

### 5.4.2 Evaluation & Discussion

We compare our models with state-of-the-art multi-document summarization systems using ROUGE and human evaluation. The former aims to evaluate informativeness and the latter targets readability.

- MCKP is the maximum coverage methods proposed by [104].
- Lin is a model that uses a class of sub-modular functions [64].
- Christ is a graph based model proposed by [16].
- M1 is our model described in Section 5.3.2. M2 is the model described in Section 5.3.3, which is resolved using an ILP method.
- MEAD [86] is a baseline that employs ranking algorithms to generate multi-document summaries.

**ROUGE Evaluation** The results are shown in Table 1. As we can see, our system (M1 and M2) produces comparable results to the state-of-the-art systems. With the MCKP method, all content words are used as concepts. But in our systems, only nouns and pronouns are regarded as entities. There are fewer nouns and pronouns than content words. This has a negative impact on the evaluation of information coverage. But according to the experiment results, our approach still obtain satisfying results based on these entities. It proves that even with much simpler feature settings of just nouns and pronouns, the proposed model generates
summarizes with good coverage of the important information in source documents. We have addressed that ROUGE is merely an index of informativeness and cannot evaluate our improvements in readability as has been proved by Christ, another coherence-focused model [16]. So we also conduct a human evaluation.

**Human Evaluation**  As some systems mentioned in Table 5.2 are not accessible, in this work we compare summaries produced by some typical systems: M2 (the best proposed system evaluated by ROUGE), MCKP (one of the state-of-the-art salience-focused methods) and humans (the gold standard).

We asked four professional annotators (who are not the authors of this part and have rich experience in annotating various NLP tasks and are fluent in English) to assign a score to each summary regarding its readability. We randomly selected 48 summaries (16+16+16) from the three systems, and asked them to assign a readability score to each document without reading the source documents (summarization is useful because we do not need to read source documents). The score is an integer between 1 (very poor) and 5 (very good). The results are shown in Table 5.1.

The average scores for the 3 systems are Human = 4.3; M2 = 3.5; MCKP = 3.1. Significance testing (significance level $\alpha = 0.05$) shows that the summaries generated by the proposed method show improvements in readability compared with previous salience-focused work.

<table>
<thead>
<tr>
<th>Type</th>
<th>SysName</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold Standard</td>
<td>Human</td>
<td>4.3</td>
</tr>
<tr>
<td>Discourse-based</td>
<td>M2</td>
<td>3.5</td>
</tr>
<tr>
<td>Maximum Coverage</td>
<td>MCKP</td>
<td>3.1</td>
</tr>
</tbody>
</table>

In our model, we assume the states of entities can be formulated as Markov chains. Although sophisticated models can be employed, such assumptions help simplify the model, and they are proved to be of use. Also, we can use more or fewer grammatical roles for entities. We tried using just two kinds of roles:
Table 5.2: ROUGE Results on DUC2004

<table>
<thead>
<tr>
<th>Type</th>
<th>SysName</th>
<th>ROUGE-1(R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Ranking</td>
<td>MEAD</td>
<td>.339</td>
</tr>
<tr>
<td>Maximum Coverage</td>
<td>MCKP</td>
<td>.385</td>
</tr>
<tr>
<td></td>
<td>ICSI</td>
<td>.384</td>
</tr>
<tr>
<td>Point Process</td>
<td>DPP</td>
<td>.398</td>
</tr>
<tr>
<td>Sub Modular</td>
<td>Lin</td>
<td>.394</td>
</tr>
<tr>
<td>Discourse-based</td>
<td>Christ</td>
<td>.373</td>
</tr>
<tr>
<td></td>
<td>M1</td>
<td>.383</td>
</tr>
<tr>
<td></td>
<td>M2</td>
<td>.390</td>
</tr>
</tbody>
</table>

presence and absence, and the performance we obtained was unsatisfying.

5.5 Related Work

A summary is much shorter than the original documents but still needs to provide readers with sufficient information. Hence the summarization systems need to identify important information and keep as much of it as possible. Most existing research follows such a guideline and takes salience as its sole focus.

Salience-focused systems cannot guarantee the readability of the generated text as they fail to take coherence into consideration. Sentence reordering, as a post processing task has begun to develop. Apparently, it cannot make up for the flaws of salience-focused systems because it is simply a reorganization of sentences. Besides, it also faces problems when dealing with temporal text [124, 30]. A better solution is to consider coherence when selecting sentences. Such comprehensive models have been proposed. Most of them are discourse driven and sacrifice informativeness for coherence. In this sense, our model is novel in dealing with coherence without discourse analysis.
CHAPTER 5. ENTITY-BASED COHERENT SUMMARIZATION

5.5.1 Salience-Focused Method

As stated, the summarization systems need to identify the important information and keep as much of it in the generated summaries as possible. One straightforward method is Maximum Marginal Relevance [14] (MMR). It is a greedy method, and is proposed to select sentences that are most relevant but not too similar to the already selected ones. It tries to keep a balance between relevance and redundancy. MMR is also widely employed to avoid redundancy in summarization systems. Among existing research, one popular kind is the ranking method (e.g., Textrank [72], Lexrank [25] and its variants [108, 112]), which construct a graph between text units and use ranking algorithms to select top sentences to build summaries. Another kind is the optimization method. Our work is one of this kind. It formulates summarization as finding a subset that optimizes certain objective functions without violating certain constraints. To find such an optimal subset is a combinatorial optimization problem, which is an NP hard problem and hence cannot be solved in linear time [71].

Recently, maximum coverage methods have been proposed and yield good results [32, 31, 104]. Maximum coverage methods formulate summarization as a maximum knapsack problem (MKMC). In MKMC methods, the meanings of sentences are believed to be made up by concepts, which usually refer to content words. And summarization involves extracting a subset of sentences that covers as many important concepts as possible without violating the length constraint. It is usually formulated as an integer linear problem. And some algorithms are proposed for obtaining approximated solutions [104, 32]. [64] design a class of sub-modular functions for document summarization. The functions they use combine two parts, encouraging the summary to be representative of the corpus, and rewarding diversity separately. Other methods that have been applied to summarization include centroid-based methods [87, 93], and minimum dominating set methods [96]. All these methods suffer in coherence.
5.5.2 Coherence-Focused Method

Sentence reordering methods are developed to correct the salience-focused models. Sentence reordering tries to generate a more coherent text by reordering its contents. Rich semantic and syntactic features are used to find a better permutation for input sentences [2, 9, 81].

The drawback to sentence reordering is obvious. The preceding sentence selection focuses solely on informativeness and totally neglects coherence. Thus it prevents the improvements expected from permutation. This is confirmed by the fact that the above methods all reports limited improvement. A consideration of coherence during sentence selection leads to new methods, and these are mainly discourse driven models. These methods are proposed based on discourse theories. Popular ones include Centering Theory, Rhetorical Structure Theory [67, 35], and Penn Discourse Tree Bank style analysis [118, 110]. Some of the summarization methods encode discourse analysis results in feature presentations together with other frequency based features for sentence selection/compression. The problem is that these discourse based features usually play secondary roles, because the models all try to improve information coverage, which are evaluated by ROUGE. And ROUGE, as is commonly known, is not sensitive to coherence.

Some others work directly on discourse analysis results, and they usually try to derive a passage from a given parse tree. The problem of summarization is regarded as finding a text $T$ so that $T = \arg\max F(T|Tr)$ for a given tree $Tr$. Here $F$ is the objective function. Early representative work of this kind includes that of [69] and that of [21]. [69] parses a document as an RST discourse tree and selects textual units according to a preference ranking derived from the tree structure to generate a summary. [21] proposes a summarization system that directly models the probability of a summary given an RST tree using a noisy-channel model.

Recently, [42] has viewed summarization as a knapsack problem on trees, and uses an integer linear problem (ILP) to formulate it. A sub tree that maximizes some objective function and obeys some given constraints is extracted from the original parse tree as the summary.

Discourse tree-based methods cannot be extended to multi-document summa-
rization. [16] propose a graph model that bypasses the tree constraints. They build a graph to represent discourse relations between sentences and then extract summaries accordingly.

It is worth mentioning that the popular neural network based discourse analysis [60, 48] provides us with an alternative way of conducting discourse analysis without traditional feature engineering. It can be used in our future work of modelling coherence using semantic relations.

5.6 Conclusion

Previous summarization methods usually focused on salience and neglected coherence. This work presents a novel summarization system that combines coherence with salience. By taking entities and links between them into consideration, our weighted longest path model successfully improves the quality of summaries. The proposed model does not require discourse analysis and hence can be applied to languages which do not enjoy plenty of ready-to-use discourse analysis tools.

Overall, the proposed system is able to generate better summaries. But we are faced with other challenges. We need to deal with the temporal/logical relations between sentences. Here, we order sentences according to the coherence scores we obtain, without considering the temporal or logical relations. There is a possibility that $S_1, S_2$ has a higher coherence score than $S_2, S_1$, but logically $S_2$ happens before $S_1$, so we have to employ $S_2, S_1$ rather than $S_1, S_2$.

Besides, in this work only syntactic linkages are used for modelling coherence. In the future, we can take advantage of the semantic relations between entities to evaluate coherence and to further improve our system.
Chapter 6

Entity-Based Memory Network

6.1 Introduction

Enabling computers to understand text as humans do has long been the goal of the natural language processing (NLP) community. A lot of NLP tasks have been tensely studied towards this goal such as information retrieval, semantic role labelling, textual entailment and so on. Among them, questions answering is of great importance and has been a huge challenge. The question answering (QA) task is to predict an answer for a given question with regard to related information. It can be formulated as a map $f: \{\text{related\_text,\ question}\} \rightarrow \{\text{answer}\}$ [59]. To predict the correct answer, computers are firstly required to “understand” the text.

Shallow features such as bag-of-words, token frequencies and so on are unable to capture the rich information in text. Often outside knowledge is required towards better performances. Traditional approaches heavily rely on rules or structured knowledge developed by experts or crowd sourcing [90, 85]. Relational databases constructed from predicate argument triples also serve as a source of knowledge [63, 97]. Problems with these approaches lie in at least two aspects. Firstly the construction of structured knowledge is both time and money consuming. Secondly it is a huge challenge to design models flexible and powerful enough to learn to employ the information extracted [39]. Thus the progress of using machine learning for QA has been slow.
Once upon a time there was a princess who lived in a high tower and she was not allowed to leave because of her mean mother. One day she chose to leave but her mother would not let her. The princess climbed out the window of the high tower and climbed down the south wall when her mother was sleeping. She wandered out a good ways. Finally she went into the forest where there are no electric poles but where there are some caves. ...

Q: Where did the princess wander to after escaping?

Figure 6.1: An Example of Question Answering

Recently the emergence of deep neural networks and distributed representations sheds light on such methods. Representing all the features using vectors provides a unified representational form for all the necessary information. Outside knowledge learnt from large corpus can be encoded into word vectors. Information obtained locally is also represented using vectors. Deep neural network models with many layers are designed to fuse information obtained from different sources [24, 46].

A notable breakthrough is to employ memories in neural networks. The representative model is named the memory network [117]. The key of memory network is to store historical sentences in a memory pool. The model is trained to look for related sentences when a question comes. Then based on the related sentences, an answer is predicted for the question. Memory network remembers all sentences it has read so that it can look for useful ones when facing questions. This model and its variants have been proved useful in a series of tasks [117, 102, 10].

One problem with memory networks is that using sentence vectors as elementary units of information makes it difficult to fully explore the information contained in text. Often is the case that in a long sentence, only part of the sentence is related to the questions. Therefore taking the whole sentence into consideration makes it hard to focus on the information that are related to questions. Besides,
6.2. MEMORIES IN DEEP NEURAL NETWORKS

learning sentence representations itself is a growing field.

We propose to focus on entities rather than sentences. Entities refer to anything that exist in reality or are purely hypothetical. We assume that text can be projected to a world of entities. The key of conducting comprehension and reasoning over text is to identify its entities and analyze the states of these entities and the relations between them. We keep a memory pool of entities and use the input sentences to update the states of these entities. Questions are answered based on the states of related entities. The proposed model deals with fine-grained information by using entities. The introduced model is named as entity-based memory network. It is tested on several datasets, including the toy bAbI dataset [116], large movie review dataset [66] and the machine comprehension test dataset [89]. Results show we have achieved satisfying results using the entity-based memory network. The rest of this chapter is organized as follows: Section 6.2 reviews some previous work. Section 6.3 describes our approaches and elaborates the details. Section 6.4 presents the experiments and the analysis. Section 6.5 concludes the chapter.

6.2 Memories in Deep Neural Networks

A lot of methods have been developed to address this problem of QA [88, 63, 11]. Recently the development of neural models leads to many NN-based question answering systems [34, 24, 46]. Among them closely related to our work is the Memory Network (MNN) [117]. The memory network contains four parts: the input module which converts sentences into vectors, the memory which keeps all sentence vectors a retrieval module and a response module. Whenever a question comes, the question is turned into a vector and the question vector is used to retrieve the memory for related sentences. The response module is used to predict an answer based on the related sentences. The core component is the memory pool that stores all the input sentences so that they can be retrieved later to answer questions. This model contains several neural networks which are jointly optimized according to the task. Experiments on a toy dataset show that this model is able to answer simple questions according to the input text. Fig. 1(a)
CHAPTER 6. ENTITY-BASED MEMORY NETWORK

(a) The Memory Network

(b) The Entity-Based Memory Network

Figure 6.2: Comparison of the Memory Network and the Entity-Based Memory Network. Sentences are decomposed into entities and then stored in the memory for later retrieval.

illustrates the memory network.

Later [56] propose the Dynamic Memory Network (DMNN) which introduces the attention mechanism into the memory network model. When retrieving memories, the location of the next related sentence is predicted according to the related sentences identified in the previous iterations. Using the attention mechanism, they obtain further improvements. Some other work [102, 10] propose other variants of MNN by introducing additional memory network modules. These work focuses on storing sentence vectors for later retrieval with no exceptions. Most of them have been tested on the toy dataset bAbI [116] and are reported to have achieved satisfying results. When further tested on some practical tasks, these
6.3 Approaches

6.3.1 Overview

Firstly we use an example to illustrate how the model works. Below we show a piece of text which contains 4 sentences and 2 questions. There are 7 entities in total, all of them underlined.

Figure 6.3: An Example from bAbI. The bAbI dataset is a toy dataset for question answering [116].

models also show the ability to produce results as good as existing state-of-the-art systems or even better results. Memory networks store sentence vectors as memories and have the superiority of processing information from a large scale. Experiment results they reported on a series of tasks are concrete proofs.

But there is also a problem with the memory networks as we have stated. Taking sentence vectors as input means that it is difficult to further analyze and take advantages of relations between smaller text units, such as entities. For example, when an entity \( e_a \) of sentence \( A \) interacts with another entity \( e_b \) of sentence \( B \), we have to take the whole sentences \( A \) and \( B \) into consideration rather than just focus on \( e_a \) and \( e_b \). This inevitably brings about noise and damages the comprehension of text. The failure of obtaining fine-grained information prevents any further improvements. In the proposed entity-based model, we focus on entities directly and avoid bringing in redundant information.
This text is elaborated around the 7 entities. It describes how their states change (i.e., the change of a character’s location) when the story goes on. Note that here all the entities are concrete concepts that exist in reality. It is also possible to talk about abstract concepts.

The core of the proposed model are entities. We take Sentence 1 ($S_1$) as input and extract the entities it contains \{Mary, bathroom\}. Vectors representing the states of these entities are initialized using some pre-learned word embeddings \{Mary, bathroom\} and stored in a memory pool. Meanwhile, we turn $S_1$ into a vector ($\vec{S}_1$) using an auto-encoder. Then we use the sentence vector $\vec{S}_1$ to update the entities’ states \{Mary, bathroom\}. The goal is to reconstruct $\vec{S}_1$ solely from \{Mary, bathroom\}. In the same way, we process the following text ($S_2$) and its entities (John, hallway) until encounter a question ($S_3$). $S_3$ is converted into a vector ($\vec{S}_3$) following the same method that processes previous input text. Then taking $\vec{S}_3$ as input, we retrieve related entities from the memory which now stores all the entities (Mary, bathroom, John, hallway) that appear before $S_3$. The related entities’ states are then used to produce a feature vector. In this case, (Mary and bathroom) are related to the question and their states are used for constructing the feature vector. Note the current states of the two entities (Mary and bathroom) are different from their initial values due to $S_1$. Based on the feature vector, we then use another neural network model to predict the answer to $S_3$.

The model monitors the entities involved in text and keeps updating their states according to the input. Whenever we have a question with regard to the text, we check the states of entities and predict an answer accordingly. The proposed model comprises of 4 modules, as is shown in Fig. 6.4. Each module is designed for a unique purpose and together they construct the entity-based memory network model.

1. I: Input module. Take as input a sentence and turn it into a vector. Meanwhile, extract all the entities it contains. The question is also processed using this module.

\(^{1}\)Note that the sentence vector is not used to answer question directly and it is also plausible to use other models to learn sentence representation.
2. G: Generalization module. Update the states of related entities according to the input. For entities that are not contained in the memory pool, create a new memory slot for each of them and initialize these slots using pre-learned word embeddings.

3. O: Output feature module. It is triggered whenever a question arrives. Retrieve related entities according to the input question and then produce an output feature vector accordingly.

4. R: Response module. Generate the response according to the output feature vector.

Figure 6.4: Architecture of the Entity-Based Memory Network. The model is divided into four modules which are shown in the figure using squares.

6.3.2 Entity-Based Memory Network Model

Here we present a formal description of the proposed model. Assume we have sentences $S_1, S_2, ..., S_n$ whose entities are annotated in advance as $e_1, e_2, ..., e_m$. 
Input Module  We firstly turn each sentence $S_i$ into its vector representation:

$$\vec{S}_i = f_1(S_i) \quad (6.1)$$

Generalization Module  For a sentence $S_i$, we collect all the entities it contains $\{e^i_1, e^i_2, \ldots, e^i_k\}$. These entities’ states $\{\vec{e}^i_k\}$ are simultaneously updated according to $\vec{S}_i$ as follows:

$$\{\vec{e}^i_k\} = \arg\min_{\{\vec{e}^i_k\}} (|\vec{S}'_i - \vec{S}_i|); \vec{S}'_i = f_2(e^i_1, \ldots, e^i_k); \quad (6.2)$$

$f_2$ is to reconstruct $\vec{S}_i$ only using the states of $S_i$’s entities $\{\vec{e}^i_k\}$. $\{\vec{e}^i_k\}$ are updated to minimize the difference between $\vec{S}'_i$ and $\vec{S}_i$. Recall that $\vec{S}_i$ is generated using $f_1$ with the whole sentence $S_i$ as input. We compress the information carried by $S_i$ into a vector $\vec{S}_i$ and then unfold it into $\{\vec{e}^i_k\}$.

After processing these sentences, we construct a memory pool which consists of entities whose states are regarded as capable of representing the information carried by the input text.

Output Feature Module  Question $q$ is turned into a vector $\vec{q} = f_1(q)$ and then $\vec{q}$ is used to retrieve related entities from the memory pool.

Figure 6.5: The Generalization Module. Using $S$ as an example, the auto-encoder is used to convert the sentence into a vector $\vec{S}$ and the entities contained in $S$ are used to reconstruct the sentence vector.
6.3. APPROACHES

Figure 6.6: The Output Feature Module. In each iteration, entities are assigned different scores which indicate their importance in constructing the output feature vector.

\[
\begin{align*}
\vec{O}_0 &= \vec{Q}_0 = \vec{q}, E_0 = \phi \\
\vec{Q}_{j-1} &= h(\vec{Q}_{j-2}, \vec{e}_{j-1}), j = 2, 3, \ldots \\
e_j &= \arg \max_{e_k \notin E_{j-1}} p(\vec{e}_k, \vec{Q}_{j-1}); E_j = E_{j-1} \cup \{e_j\} \\
\vec{O}_j &= u(\vec{O}_{j-1}, p(\vec{e}_j, \vec{Q}_{j-1}) \ast \vec{e}_j)
\end{align*}
\] (6.3)

At first, \(\vec{Q}\) is initialized using \(q\). In the \(j\)th iteration, \(p(\vec{e}_k, \vec{Q}_{j-1})\) is the probability (or score) of \(e_k\) being selected to compose the feature vector for answering \(q\). Note that every \(e\) is considered only once. In \(\vec{Q}\), we consider the entity selected in the previous iteration. \(\vec{Q}\) is kept updated using \(e\) and \(p\).

After several iterations, we use the final \(\vec{O}_m\) as the output feature vector \(\vec{O}\). Note that if the \(\vec{O}\) does not change much between iterations, we will omit the remaining loops. This early-stop strategy helps reduce the time cost.

**Response Module** Then we decide the answer using \(a(q) = v(\vec{O})\). \(a(q)\) produces a vector whose each item corresponds to one word in the vocabulary. \(a(q)_i\) indicates the probability of \(word_i\) being used as the correct answer. We choose the one with the highest probability. Models like recurrent neural network can be used to output a sentence as the answer.
6.3.3 Implementation

This is a supervised model and requires annotated data for the training. The training data contains the input text, questions and answers. Also we need all the entities and entities that are related to the answer labelled.

We define the function form for training as follows: As for $f_1$, many models, like the recurrent neural network, recursive neural network and so on [74, 98, 58], can be used to convert a sentence into a vector. Here we use an Long Short-Term Memory (LSTM) auto-encoder [61] which takes a word sequence as input and outputs the same sequence.

$f_2$ takes a list of entity states as input and tries to reconstruct $\vec{S}_i$. We use the Gated Recurrent Unit (GRU) [17].

$$\vec{S}_i^k = \tanh(GRU(S_i^{k-1}, e_i^k))$$
$$\vec{S}_i = \vec{S}_i^3$$

(6.4)

A GRU can be represented as the follows:

$$\begin{align*}
z_t^j &= \delta(W_z \ast x_t + U_z \ast h_{t-1})^j \\
\vec{h}_t^j &= \tanh(W \ast x_t + U \ast (r_t \circ h_{t-1}))^j \\
r_t^j &= \delta(W_r \ast x_t + U_r \ast h_{t-1}) \\
h_t^j &= (1 - z_t^j)h_{t-1}^j + z_t^j\vec{h}_t^j
\end{align*}$$

(6.5)

$\circ$ represents an element-wise multiplication. $z_t^j$ and $r_t^j$ are two gates controlling the impact of historical $h_{t-1}^j$ on the current $h_t^j$. The GRU takes $x_t$ as input and updates the state of the neuron to $h_t^j$. Compared with LSTM which it often replaces, it simplifies the computation while still keeps a memory of previous states. Therefore it takes less time to train GRU than LSTM.

Our goal is to minimize the loss $|\vec{S}_i - \vec{S}_i|$. Using the stochastic gradient descent, we are able to train $f_2$ and also update $\{e_i^k\}$. Note that the input module and the generalization module do not interact with the remaining. Thus they can be trained in advance.

The output feature module checks the memory pool repeatedly to select enti-
ties to form a feature vector:

\[ \begin{align*}
Q_{j-1} &= \tanh(GRU(Q_{j-2}, e_{j-1})) \\
e_j &= \arg \max p(e_j, Q_{j-1}) = \arg \max \text{sigmoid}(W * GRU(e_j, Q_{j-1}) + b) \\
\tilde{O}_j &= \tanh(GRU(O_{j-1}, p(e_j, Q_{j-1}) * e_j))
\end{align*} \]

(6.6)

To generate the final answer, we use a simple neural network which takes the feature vector \( \tilde{O} \) as input and predict a word as output. \( p_w = v(\tilde{O}) = \text{softmax}(tanh(W' * \tilde{O} + b)) \). The word with the highest probability is selected. Suppose a sentence is to be generated, we use the GRU to update \( \tilde{O} \) and then generate the sentence \( \{w_*\} \) as follows:

\[ \begin{align*}
\tilde{p}_w^{-1} &= \text{softmax}(tanh(W' * \tilde{O}_{i-1} + b)) \\
w_{i-1} &= \arg \max \tilde{p}_w^{-1} \\
\tilde{O}_i &= \tanh(GRU(\tilde{O}_{i-1}, w_{i-1}))
\end{align*} \]

(6.7)

Similar to \([117]\), we use the stochastic gradient descent algorithm to minimize the loss function shown in Equation (6.8) over parameters. For an input \( S_i \) and a given question \( q \) annotated with the correct answer word \( a \) and related entities \( \{e_r\} \), the loss function is as follows:

\[ \sum_{i \neq r} \max(0, \gamma - (p(e_r, q) - p(e_i, q))) + \sum_{l \neq a} \max(0, \gamma - (p_{\text{word}_a} - p_{\text{word}_l})) + ||\Theta||^2 \]

(6.8)

\( \gamma \) is the margin and \(||\Theta||^2 \) is the squared sum of all parameters which is used for regularization. Note that \( \Theta \) does not include parameters of \( f_1 \) and \( f_2 \). Their parameters and states of entities are learned as described in Section 6.2.2. Word vectors used to initialize entity states and words in auto-encoder come from GloVe \([83]\). The dimension is set to be 50.

### 6.3.4 Data Preparation

The model requires entities to be annotated in advance. In this work, we treat each noun and pronoun as an entity. Different words are regarded as different entities for simplicity. This strategy saves us the effort of entity resolution which
is a challenge for many languages. It also makes possible the application of the proposed model to entity resolution\(^2\). For datasets with related entities annotated, we can use the loss function described above. But annotating the related entities is time and labour-costing. Most datasets available are not annotated. The weakly supervised learning can be applied to such data by trimming the loss function to

\[
\sum_{l \neq k} \max(0, \gamma - (p_{word_k} - p_{word_l})) + ||\Theta||^2
\]  

For unannotated data, a fully supervised training is also possible if we regard entities contained in questions as related entities or if we can use other methods to identify entities that are believed to be related.

### 6.4 Experiments

To verify the effectiveness of the proposed model, we conduct experiments on several datasets, including a toy QA data set bAbI [116], the large movie review dataset for sentiment classification [66] and the Machine Comprehension dataset (MC Test) [89].

#### 6.4.1 bAbI Dataset

The example shown in Fig. 1 is extracted from the bAbI dataset. It contains 20 topics, each of which contains short stories, simple questions with regard to the stories and answers. The data is generated with a simulation which behaves like a classic text adventure game. According to some pre-set rules, stories are generated in a controlled context.

Previous work reports satisfying results using memory networks for most topics (around 90% for most of them). However, we notice an interesting thing that all of them with no exception fail on the problem of path-finding which is to predict a simple path like “north, west” given the locations of several subjects. Another one is the positional reasoning. The Memory Network [116] reports accuracies

\(^2\)We treat each mentions of entities as different one when processing the text and ask questions about which of these mentions refer to the same entities.
6.4. EXPERIMENTS

of 36% and 65% for the two topics. The Dynamic Memory Network [56] reports
accuracies of 35% and 60%. The proposed model (Entity-MNN) reports accuracies
of 53% and 67% respectively. It is still far from satisfying but the improvements
on the two tasks indicates the superiority of the entity-based memory network.
For the whole dataset, we report mean error rates about 12%, comparable to 3.2
to about 24 reported by previous work [102, 56, 116].

The data is generated in a controlled text. As we know, QA systems trained on
controlled text normally suffers when moving to real world problems [39]. Results
on this toy dataset is not as convincing as that on practical tasks. Given how
the bAbI data is generated, it is easy to achieve a 100% accuracy if we do simple
reverse engineering to identify the entities and rules. The good results of memory
networks, including our model, can not be solely attributed to their ability of
comprehension. It may be partly due to their ability of inducting the entities and
rules from text.

6.4.2 Machine Comprehension Test Dataset

We tested the proposed model on a dataset constructed from children stories. The
machine comprehension test (MCTest) dataset [89] has 500 stories and 2000 ques-
tions (MC500). All of them are multiple choice reading comprehension questions.
An additional smaller dataset with 160 stories and 640 questions (MC160) is also
included in the MCTest data and used in our work.

Since the proposed model does not consider the form of multiple choice ques-
tions, we need to convert MCTest data into suitable formats firstly. When an-
swering a multiple choice question, one is provided with several alternatives of
which at least one is correct. These alternatives can be regarded as information
known.

For a question, we replace the “Wh-” words using each alternative and Each
alternative is turned to a new declarative sentences. These generated declarative
sentences are generally understandable though may not be grammatically correct.
Then we use the proposed system to decide whether the generated sentences are
correct or wrong. However, we do not distinguish between questions with only one
answer and those with more than one answers as these newly generated sentences
Table 6.1: Results on Machine Comprehension Test Dataset

<table>
<thead>
<tr>
<th>Sys.</th>
<th>Acc. (%) MC160</th>
<th>Acc. (%) MC500</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single</td>
<td>Multiple</td>
</tr>
<tr>
<td>Richardson’13 [89]</td>
<td>76.8</td>
<td>62.5</td>
</tr>
<tr>
<td>Wang’15 [109]</td>
<td>84.2</td>
<td>67.9</td>
</tr>
<tr>
<td>Sachan’16 [92]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EntityMNN</td>
<td>Average=76.1</td>
<td></td>
</tr>
</tbody>
</table>

are treated separately. In other words, all questions are treated as having multiple answers.

The MCTest contains only hundreds of stories and is usually used for test only as statistical models normally require a large amount of training data. However, we still obtain satisfying results using this dataset. Table 6.1 demonstrates the effectiveness of the entity-based model on the MCTest dataset. We outperform the previous state-of-the-art [109, 92] on both MC160 and MC500. Our model does not employ rich semantic features as others do, and hence is easy to be migrated to languages aside from English.

6.4.3 Large Movie Review Dataset

We further tested our model on the Large Movie Review Dataset [66], which is a collection of 50,000 reviews from IMDB, about 30 reviews per movie. Each review is assigned a score from 1 (very negative) to 10 (very positive). The ratio of positive samples to negative samples is 50:50. Following the previous work [66], we only consider polarized samples with scores no greater than 4 or no smaller than 7.

For each review, we present it as a short story and then add a question “what is the opinion?”. The answer is either “negative” or “positive”. In this way we turn this task into a question answering problem. Note that although here the answer to a question is either “negative” or “positive”, we do not put any constraints on the output. It is treated in the same way as open domain question answering and the system is expected to learn to predict the output by itself.
Table 6.2: Results on Large Movie Dataset

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc.</td>
<td>89</td>
<td>93.4</td>
<td>95</td>
<td>97.2</td>
</tr>
</tbody>
</table>

We do not use the full dataset as the training takes a long time. We randomly select 10K samples (5K negative + 5K positive) for training and another 10K for test. We obtain an accuracy of 97.2% on the subset which is higher than previous work [66, 50, 51] as is shown in Table 6.2. By exploring relations between entities, we consider information that is usually not included for classification tasks and obtain better results.

6.4.4 Analysis

The proposed model is designed based on the assumption that entities are the core of text. By updating the states of entities, information carried by text is encoded into entities. Thus all questions which are related to the text can be answered based on entities solely.

Using entities enable us to break a sentence into smaller text units and analyze text from a smaller scale. As stated, if one entity $e_i$ in sentence $S_a$ interacts with another entity $e_j$ in sentence $S_b$, dealing with $e_i$ and $e_j$ directly is much easier than dealing with $S_a$ and $S_b$. Therefore the proposed model overcomes this problem as has been proven in our experiments. A shortcoming with the proposed model is that, it cannot handle text that contains very few entities. Also hidden entities are not considered. As we know, pro-drop languages, like Japanese and Chinese, tend to omit certain classes of pronouns when they are inferable. The proposed model will encounter problems when dealing with such text.

In this work, we focus on QA but also try to resolve other problems like classification using the same model.

QA is an important task in natural language processing (NLP). It is an efficient way of obtaining information from text and a natural way of interacting with computers. More than that, almost all problems in NLP can be formulated as QA tasks. Some ones, like information retrieval and dialogue system, are by nature
CHAPTER 6. ENTITY-BASED MEMORY NETWORK

question answering tasks. Other problems, like machine translation, POS tagging, co-reference resolution and so on, can also be formulated as question answering tasks. Take the co-reference resolution for example, given a piece of text, we raise questions like “What does XX refer to?” and expect the system to give correct answers. Similarly, we can model POS tagging as a question answering task by asking “What are the parts of speech?”.

Formulating these tasks as QA provides us with the convenience of solving several different problems using one scheme. This point is of vital importance. As we know, existing work on different NLP tasks are highly differentiated, each designed for an (or a class of) unique task(s) with unique features and unique architectures. It is almost impossible to develop a comprehensive system which can conduct several different tasks without damaging the performance.

Developing a comprehensive system with a unified scheme faces challenges in feature representations and model design. Feature representation converts text into features which can be easily computed by models. Models are designed accordingly to process the input features and generate the desired output. A comprehensive model requires us to develop feature representations which are capable of storing all the information contained in text as different tasks may need different information, and to develop a model which is capable of paying attention to different aspects of the information carried by features with regard to the problems raised and generating the desired results. The two are the core challenges to be met and overcome towards a comprehensive system.

Although challenging, such comprehensive systems are of great interest to the artificial intelligence community in their ability of comprehension. Formulating different tasks as QA problems and resolving them using a unified scheme is to some extent, closer to how humans process languages. It differs from previous work in that comprehension of text is needed to serve as the basis for answering various questions.

A model could be regarded as capable of comprehension, if it was able to answer questions raised from different aspects with regards to the input text.

All deep learning models rely on distributed representations representing various features as vectors. These vectors are believed to have encoded all the se-
mantic and syntactic information in themselves. By replacing the various features used in traditional models with vector representations, we can resolve the problem of feature representations. But existing deep neural network models are often developed for a certain problem or a certain class of problems. In other words, they are in no sense different from traditional methods in being highly differentiated. Now with the distributed representations and deep neural networks, it becomes probable to develop such a multi-purpose system in a unified scheme. Text is represented as vectors and processed by various neural models.

6.5 Conclusion

This work presents the entity-based memory network model for text comprehension. All the information conveyed by text is encoded into the states of entities it contains and questions regarded to the text are answered using these entities. Experiments on several tasks have proven the effectiveness of the proposed model. The proposed model is based on the assumption that entities can express all the information of text. In future research, we will further explore its ability by considering more components in text.
Chapter 7

Entity-Based Text Representation

7.1 Introduction

Text representation is one of the foremost problems in natural language processing. A good representation which reserves the syntactic and semantic relations of text will bring great benefits for NLP tasks.

The prevalent method of text representation is the distributed representation [76] which turns text (word, phrase, sentence, document,...) into dense, low dimensional vectors. As stated in previous chapters, information contained in text, semantic as well as syntactic information, is encoded into vectors. This enables computers to perform tasks which used to be difficult, like measuring the similarities between text, representing logistical relations and reasoning.

Vector representations also save the effort of designing feature templates, since we have encoded all the information into vectors. There is no need to design feature templates and extract features. The unified representation of features also helps simplify the model design as the inputs always take the form of vectors. Recent work based on the distributed word representations have achieved satisfying results for various tasks [74, 20, 1].

But problems emerge as researchers start applying the vector representations to long text like sentences or documents [58]. As we know, vector representations
are initially designed for words [76]. The embedding of a word is learnt according to the word’s neighbours in text. The easy access to large corpus makes the supervised learning algorithms like skip-gram, negative sampling possible.

The situation changes completely when it comes to larger text units. These models cannot be well applied. Unlike words, even a very large corpora cannot provide enough data for models to learn the co-occurrence information for phrases and sentences, let alone paragraphs and documents. Thus learning vector representations for large text units directly becomes impossible.

To bypass the data sparsity problem, quite a few composition-based methods have been proposed for inferring representations for large text units using the words they contain [58, 94, 43], given the fact that phrases/paragraphs/documents are all comprised of words.

Currently the recurrent neural networks [94] are the dominating models for learning representation for long text. It is assumed a piece of long text is generated by assembling words one by one. For each time step, one word is taken in and the text representation is updated accordingly. This procedure is repeated until all the words in text are taken into consideration.

RNN and its variants are widely adopted for representing long text, usually sentences. Sentence-level NLP tasks thus enjoys rapid development as the word-level tasks did. But the improvements we obtain for sentence-level tasks are not as significant as that on word-level tasks. The problems lie in several aspects. Among them one is the gradient vanishing/exploding. If a small value got multiplied for many times, it may become intractable. Models like LSTM [43], GRU [17], and residual network [37] are proposed to address this problem from different aspects.

Another problem lies in the model itself. Similar to word representations, these models try to compress the sentence into a dense vector. But a sentence usually contains more information than a word does. One vector may be enough to represent the information of a word but is not enough for a sentence or a paragraph. In this case, additional storage helps for better representation for the text. One example of using additional storage for text representation is the attention mechanism [57, 23] which is widely practised now. The attention mechanism enables the model to look at the original input again after obtaining the fixed-length
vector. It learns to attend to different parts of the original input according to the context. As a plenty of work [1, 120, 80] verifies, the additional information provided by an extra attention layer enhances the vector representation generated by recurrent models. It is widely believed the attention layer can be trained to attend to the most relevant and specific input as human brains do.

Inspired by the attention mechanism, we propose a novel method to enhance the recurrent model for learning text representation by considering more additional information. We take advantages of the entities contained in text. These entities, are regarded as the core of text and are used to represent all the information contained in the original input.

As stated in previous chapters, we assume that text nonetheless entails some entities. All the components in text center on these entities. Text talks about the states of these entities and the relations between entities. To understand a piece of text is to know the states of these entities and the associated relations. Take the following sentence as an example. “I ate an apple.” This sentence contains two entities “I” and “apple”. The relation between the two entities is “A ate B”. We can thus roughly decompose the sentence into two parts: the entities it contains, “I” and “apple”, and the relation between these entities “A ate B”.

Such a distinction between entities and the relation is reasonable and meaningful. Entities, as we know, are something that exist and what the text is talking about. The relation, is about the states of entities, how they interact with each other and how the states of entities change. Entities are concrete and do not change much. Relations, on the other hand, are abstract and change often. For each entity we use a vector to represent its state. And an additional vector is used to represent the relations between entities. In this way, we separate the concrete parts from the abstract parts of sentences. “Render unto entities the things that are entities, and unto relations the things that are relations.”

In our model, we do not deal with sentences which contain too much information for a vector to carry. Instead we use its components. For a sentence with \( n \) entities, we will have \( n + 1 \) vectors which represent the \( n \) entities and the relation. This strategy, though can be used for a variety of NLP tasks, proves most effective for machine translation. In machine translation, when mapping sentences
from the source language to the target language, usually the change in structure is dramatic. This is especially the case when translating between two languages without much relation, like English-Japanese. The huge change in sentence structure is one of the many factors that make machine translation challenging. But using the entity-based text representations, we find that the translation can be divided into two types: translation of entities and translation of entities. As we know, the translation for entities rarely changes. For example, for a sentence pair “I ate an apple.” “私は林檎を食べた。” , from a sequence-to-sequence view, the mapping between words and positions is confusing and hard to model. But the correspondence between entities such as “I-私”, and “apple-林檎” are clear and relatively easy to deal with. The mapping between entities can be regarded as the anchors during translation. The difficult lies in the translation of relations: “A ate B”-“A は B を食べた”. If we can separate the entities from the relations, we are able to generate high quality entity translation. Also, we can focus on using various resources improving the relation translation and entity translation.

Figure 7.1: An Example of the Entity-Based Machine Translation
7.2. **ENTITY-BASED TEXT REPRESENTATION**

In this work, we propose to firstly use an encoding model to decompose a sentence into a set of entity vectors and a relation vector. Then a translation model maps entities vectors and relations vectors from the source language to the target language. A decoding model is used to translate the newly obtained vectors into a target sentence. The same strategy can be used for other tasks such as classification and question answering. All that we need to do is to replace the translation and decoding model with a classification layer or RNN layer to generate the desired task-specific outputs.

In the following section, we will show the details of the proposed model. Section 7.3 presents the experiment results. Section 7.4 reviews some related work and Section 7.5 concludes this chapter.

### 7.2 Entity-Based Text Representation

In this section we show how to represent text using entities it contains. Given a sentence with \( N \) entities, we will have \( N + 1 \) vectors of which the \( N \) vectors correspond to the \( N \) entities and the additional one is the relation vector. These entity vectors are initialized using the pre-trained word embeddings and the relation vector is randomly initialized. We will then update these vectors so as to encode information carried by input into these vectors. The details will be elaborated below.

For a sentence of the source language, \( S^s = \{w^s_0, w^s_1, w^s_2, \ldots, w^s_{n-1}\} \), we firstly extract all the entities \( e^s_0, e^s_1, \ldots, e^s_k \in S^s \). \( r^s \) represents the relations between entities. All the information carried by the sentence will be passed to the entity and relation vectors.

A bi-LSTM model [33] is used to scan the entities and the relation vector to generate an intermediate representation \( \tilde{h} \). Then we try to generate the \( S^s \) based on \( \tilde{h} \). Entity vectors and the relation vector are updated accordingly to minimize the loss. This procedure is repeated several times until the loss converges. The algorithm we used is shown in Algorithm 2.

Note this section is independent from the rest thus can be easily adopted by tasks besides the machine translation as we mentioned above. It is also plausible
Algorithm 2 Entity-Based Text Encoding Algorithm

Input: \( S^s = \{w_0^s, w_1^s, w_2^s, ..., w_{n-1}^s\}, e_1^s, e_2^s, ..., e_k^s \in S^s \).

\( \vec{e}_i^s \) is initialized using the vector representation of entity \( e_i \)

\( r^s \leftarrow \vec{0} \)

repeat

\( \vec{h} = BiLSTM(e_1^s, e_2^s, ..., e_k^s, r^s) \)

output = \( \{y_0, y_1, ..., \} = RNN(\vec{h}) \)

loss = \( \text{cross entropy}(\text{output}, S^s) \)

\( e_1^s, e_2^s, ..., e_k^s, r^s = \arg\min \ \text{loss} \)

until loss converges

Output \( e_1^s, e_2^s, ..., e_k^s, r^s \)

---

Figure 7.2: The Entity-Based Encoding Module

---

to change it to an end-to-end model. But an end-to-end model needs to be trained every time facing a new task. And as we know, the training takes a long time, especially for a large corpus. Another problem with end-to-end model lies in
7.3. NEURAL TRANSFER-BASED MACHINE TRANSLATION

generalization. End-to-end models normally work well on corpora similar to the training datasets but can hardly be directly transferred to different datasets.

7.3 Neural Transfer-Based Machine Translation

Existing machine translation systems are mainly sentence-based which take sentences as input and output. The source sentence is encoded into a vector and then the model decodes the vector into the target sentence. As we stated above, one vector is not enough for representing a sentence. The state-of-the-art systems usually use the attention mechanism which offers a chance for the decoding model to acquaint itself with the input again. Fig. 7.3 shows a diagram of the sequence-to-sequence with attention mechanism machine translation model [1].

The proposed transfer-based model also uses sentences as the basic text unit. The difference lies in the intermediate representations of sentences. We use several vectors to represent one sentence. The flowchart of the proposed system is shown in Fig. 7.4.

7.3.1 Encoding

The encoding part is exactly what is described in the above part. During the training phase, we extract entities from a sentence pair \( \{S_s, S_t\} \), and update the corresponding entity representations according to the above model.

The updated entity vectors \( \vec{e}_i^s \) and the new relation vector \( \vec{r}^s \) learnt from the source sentence \( S^s \), are passed to the translation module. Note the vectors now have been updated to contain information of \( S^s \) and are different from their original states.

For the target sentence \( S^t \), similarly, we have vectors \( \vec{e}_i^t \) and \( \vec{r}^t \). They are also passed to the next module for supervised learning.

**Alignment** During the training, we need to align entities of \( S^s \) to entities in \( S_t \). Some entities only appear in the source sentence or the target sentence \(^1\). We

\(^1\)This linguistic phenomena is related to zero-noun or empty category. see Chapter 4 for details.
will add additional vectors to represent their counterparts which are hidden in the other sentence. Adding new entities will cause redundancy but losing entities will result in loss of information. We employ such a strategy because we believe the neural network can handle the redundancy better than incompleteness of
Figure 7.4: The Entity to Entity Machine Translation Model

During the test, since we have only one sentence, there is no need to align entities.
7.3.2 Translation

This module is used to map vectors from one space to another.

Here entities and relations are treated separately. As we know, during the translation, the mapping between entities fluctuates less. If the source sentence talks about an apple, the target sentence should also talk about the same entity "林檎".

\[
\begin{align*}
\vec{e}_t &= f_1(\vec{e}_s) = \text{ReLU}(W_1 \vec{e}_s + b_1) \\
\vec{r}_t &= f_2(\vec{r}_s) = \text{ReLU}(W_2 \vec{e}_s + b_2) \\
\vec{h}_t &= \text{RNN}(\vec{e}_s, \vec{r}_t)
\end{align*}
\]

The incoming vectors are \(\vec{e}_s\) and \(\vec{r}_s\) which are learnt from the source sentence and the out-coming vectors are \(\vec{e}_t\) and \(\vec{r}_t\) which are believed to be the corresponding vectors in the target sentence. A good model should generate \(\vec{e}_t\) and \(\vec{r}_t\) with the same values as \(\vec{e}_t'\) and \(\vec{r}_t'\).

7.3.3 Decoding

The decoding part can be regarded as the inverse process of the encoding part. In this work, we employ an additional attention layer which has been proved useful in a handful of previous work.

\[
\begin{align*}
a_{w_t} &= f_3(\vec{w}_s, \vec{w}_{i-1}, \vec{h}_t) \\
&= W'_3(\text{ReLU}(W_3[\vec{w}_s : \vec{w}_{i-1} : \vec{h}_t] + b_3)) + b'_3 \\
p_{w_t} &= f_3(\vec{h}_t, \sum a_{w_t} w_s) \\
&= \text{Softmax}(W'_4(\text{ReLU}(W_4[\vec{h}_t : \sum a_{w_t} w_s] + b_4)) + b'_4) \\
w_t &= \arg \max p_{w_t} 
\end{align*}
\]

A sentence of the target language is generated using the above equation. We select the word with the highest probability for each step. The loss is calculated using the softmax cross entropy by comparing \(p_{w_t}\) with \(w_{t}^{*} \in S^t\).
7.4. EXPERIMENTS

The decoding part is similar to existing sequence-to-sequence model. When generating the output, an attention layer is used to scan the input again to help improve the quality of the generated text.

This is the entity-to-entity model for machine translation. For other tasks, say the classification, all that we have to do is to replace the Translation and Decoding by a classification module.

7.4 Experiments

To verify the performances of the proposed model, we conduct experiment on the English-Japanese dataset. The details are elaborated below.

7.4.1 Machine Translation

Data Preparation

We use the English-Japanese legal documents for our experiment. The legal documents contain lines that are formal and precise. They contain lots of entities which is very suitable for our model. We remove sentences that are too long or too short and keep about 260k parallel sentences. The English and Japanese word embeddings are trained on the Wikipedia with a dimension of 300. We keep 40k tokens and replace those whose frequencies are too low using “UNK”. The data is divided into three parts, the training dataset (80%), the development dataset (10%) and the test dataset (10%).

In this experiment, we try to translate English sentences into Japanese sentences. The source language is English and the target language is Japanese.

Training

The training involves three steps.

1) Updating entity vectors and relation vectors.

During the first step, we need to extract all the entities contained in sentences, both the source sentences and the target sentences. Then we update the entity
vectors and the relation vectors by forcing them to generate the original source and target sentences.

We also need to align these entities for the training in the second step.

2) Mapping entity vectors and relation vectors from the source language space to the target language space.
An IBM1 model is trained using the parallel sentence pairs. This model is used to decide the alignment between English entities and Japanese entities. We then tag both the source sentences [6] and the target sentences [55]. All the nominal words are extracted and aligned. For entities that only appear in one side, we add the corresponding ones in the other sentence. The entity representations are initialized using pre-trained word vectors and updated using the method described in previous section. After obtaining the updated entity vectors and the relation vector, we are able to train a mapping model which projects entity vectors and relation vectors from the English vector space to the Japanese vector space.

3) Generating the target sentences based on the translated entity vectors and relation vectors.
Using the translated entity vectors and the relation vector, a Japanese sentence is generated accordingly. During the training phase, we use the Japanese entity vectors and Japanese relation vectors learnt in Step1 as input to generate the Japanese sentence. It is similar to the procedure in Step 1. Here we use the attention mechanism, which looks at the English sentence when generating the output.

Test

The testing process also involves three steps.

Step 1 is the same with training except we only have the source sentence here. Hence we only need to update the entity vectors and relation vectors for English sentences. Step 2 is to map the learnt vectors from one space to another. In step 3, we use the English sentence and the vectors obtained in Step 2 to generate a Japanese sentence as the translation.
Table 7.1: Results on En-Ja Legal Documents Dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNNSearch</td>
<td>17.2</td>
</tr>
<tr>
<td>Proposed</td>
<td>19.0</td>
</tr>
</tbody>
</table>

**Evaluation**

We implemented a sequence-to-sequence-with-attention model as the baseline. This is also the model adopted by some state-of-the-art systems [1]. In order to make the results comparable, we follow the same preprocessing procedures and use the same hyper-parameters. We evaluate the results using BLEU as shown in Table 7.1.

**7.4.2 Analysis**

The key of neural machine translation is to represent text as vectors. The source sentence is encoded into the vector and the target sentence is to be decoded from the vector. The vector should completely preserve the information. This is also the crux of many other neural models. But the existing recurrent models, as we stated above, cannot produce presentations good enough for practical applications. The attention mechanism helps improve the quality of text representation by using additional vectors. In the proposed model, we give up the idea of representing a whole sentence using one vector. We decompose a sentence into entities and relation. We use one vector for one entity and use one additional vector to represent the sentence. Compared with the existing models, we have more vectors to represent the text. Thus we are able to keep more information.

Another point which matters is that the proposed model involves three steps. The first step is to encode information of the source sentence into vectors. The second step maps the vectors from the source language space to the target language space. The last step generates the output from translated vectors. The first and the last steps are about encoding and decoding in the same language space. The second step is about the translation. During the second step, we have entities which can be aligned using outside knowledge. The only difficulty lies in the
relation translation. Compared with the sequence-to-sequence model which uses only two steps, encoding and decoding, the proposed model have different parts focused on different purposes. We do not need to squash various information into one model and one vector.

Experiments show that the proposed model shows improvements in the quality of machine translation. We produce results better than the existing state-of-the-art model under the same circumstance.

7.5 Text Representation

Efficiently representing a piece of text has long been a challenging problem for NLP research. A naive method is the bag of words representation. Other useful features such as tf-idf, N-grams, parts-of-speech and so on have also been explored. These conventional text representation methods are widely used but there exists several problems. Firstly it is labour-costing to design such features. For different tasks, different features play different roles. Hence we need task-specific features. More than that, different models take distinct forms of features, so these features are not only task-specific but also have model-specific forms. Secondly, some important features cannot be easily well-formed, such as the relations between words. With the distributed representations [40], these problems can be overcome easily. All features are presented as vectors and all neural network models take vectors as input. Besides, the semantics, which has long been believed vital to natural language processing, but hard to calculate can be represented using these vectors.

Despite its advantages, the distributed representation also has its own problems. For word representations, we can easily obtain the vectors using a huge corpus [73, 83]. While due to data sparsity, the same methods for generating word representations can not be directly applied to get phrase/sentence level representation. The recurrent neural network [74] is used for obtaining representations for text with more than one words. The idea is straight-forward. For a piece of text $S = \{w_0, w_1, ..., w_n\}$, we have
\[
\begin{cases}
\vec{S}_0 = \vec{0} \\
\vec{S}_i = g(W[w_{i-1} : S_{i-1}] + b)
\end{cases}
\] (7.3)

The final \(\vec{S}_n\) is used as the representation for \(S\).

The recurrent network is able to handle inputs of various lengths. It is widely used in end-to-end models which means the RNN model needs to learn the parameters according to the training data. A variant worth mentioning is the recursive neural network which considers the relations between words by employing parse trees. When dealing with sequences that are too long, the recurrent and recursive model sometime will encounter the gradient exploding/vanishing problem.

The LSTM and GRU models [43, 17] are thus proposed to remedy this problem. LSTM and GRU use internal memories in the neurons to keep a record of the historical input and use gates to control the influence of old and new information. LSTM and GRU greatly improve the quality of generated text representations and have been widely adopted by various tasks. Researchers also find that by looking at the input twice, from head to rear and reversely could also improve the performances of LSTM. This is the bidirectional-LSTM (Bi-LSTM) [95], which has also been used in our work.

Another model which draws much attention recently is the residual network [37]. The residual network has a residual connection between the input and the output which allows information carried by input be passed to the final layer directly. It is also said to be able to handle the gradient vanishing/exploding problem as the loss can be passed back directly without many intermediate layers.

Convolution neural network [54] is another kind of neural networks that deals with inputs of various lengths. It employs a window of fixed length to scan the input and each time takes a fixed length of data as input. Contrast to the recurrent model which has as many steps as the length of the input, the convolution model has predefined number of layers. Thus it suffers less from the gradient vanishing/exploding problem and shows advantages in some tasks.

Above models are usually used for sentences, sometimes applied to phrases. It is possible but implausible to use them for paragraphs or documents which
contains more words and more complex structures. Dealing with even longer text is a challenge huger than that with sentences. However researchers still make several attempts. One of which is the memory network [117]. The memory network keeps a record of all previous inputs and store them in a memory pool. The memory pool is scanned through when producing the output. The memory network proves successful for understanding short, simple, computer-generated toy text.

The entity-based memory network [111], which uses entity vectors to replace sentence vectors to composite the memory pool, is developed to meet the requirements of understanding more complex text, like children stories. Experiment results on several datasets prove the effectiveness of the entity-based model. This work applies similar strategies to sentence processing and also obtains improvements.

7.6 Conclusion

We propose the entity-based text representation model and evaluate it on the task of machine translation by using an entity-to-entity attentional neural machine translation model. It differs from existing neural models in that we decompose sentences into entities and relations. The translation is then conducted at the entity and relation level. This helps us avoid compressing all the information in one vector. Experiments proves the effectiveness of the proposed model.

As stated, the key of the proposed model lies in the translation of relations. Now we only use a simple multi-layer neural network to translate relations. In the following work, we will explore more sophisticated models for relation translation.

Besides, the proposed model involves encoding and decoding using entities. In this work, we employ a recurrent model for the encoding and decoding. Using recursive models makes it possible to consider more detailed relations between entities. We will further explore the relations between entities and build a hierarchical structure to organize entities.
Chapter 8

Conclusion

In this chapter, we briefly conclude our work and the guidelines we follow. Problems addressed in this thesis are of a broad category, including learning word embeddings, detecting zero nouns, summarizing documents and representing text. These tasks are among the most challenging ones in NLP and are associated with the key problems of discourse analysis, i.e., what are the components of discourse and how are they organized together. Only a thorough analysis of discourse would eventually lead to perfect solutions to these problems. Though far from this goal currently we are, the deep analysis enables us to obtain a limited understanding of discourse. Hence we are able to employ deep information towards better systems for practical tasks.

We regard discourse as motivated by entities. A piece of discourse develops with entities as its centre. We know that a piece of discourse corresponds to an idea inside our minds. The ideas in our minds, on the other hand, comes from interactions with the real world. Everything we talk/think about originates in the nature or the observable world. Some words, like “ghost” and “thought” that come from fiction or are totally abstract, do not have physical entities associated with them. But still they are the reflection of external world inside our minds. In the observable world, entities are concrete and construct the basis. Other concepts, rely on entities. Like the relation “have” which means ownership, it can only be abstracted from a collection of scenarios that relate to ownership, such as “I have a dog”, “He has a cat”, “We have a house”, ...
Since entities are the basis of the real world, we assume they are also the basis in discourse. Entities inside our minds can be projected to entities in the real world. A piece of discourse is constructed using entities and the relations between them. To achieve understanding by observing these entities, we need to store all the information in entities. In addition to the concepts established by the people in long-term social and historical life practices, entities have to express some other information which is temporary and particular. In this work, we use vectors to represent entities. The vector learnt from a large corpus is believed to have encoded the concept of the entity and the vector learnt from a certain sentence further encode more information expressed in the sentence.

With the entity-based view we come to see the meaning in discourse in a new light. Meaning is a sum of entities and their mutual relations. To build meaning from discourse is to collect entities and describe their relations. We apply this novel view on practical applications.

Utilizing entities and their mutual relations, we develop a graph-based model for summarization. When extracting sentences to form a summary, two factors weigh heavily: 1) whether the extracted sentence is important or not, 2) whether the extracted text is well organized or not. The two aspects are usually dealt with independently. The first one is addressed according to the words that one sentence contains and the latter is evaluated according to the discourse relations between sentences. We propose to use entities to resolve both problems. Entities not only represent information but also maintain coherence in text. The objective function we propose considers both informativeness and coherence. This leads to our summarization algorithm which is effective and easy to adopt for different languages.

More than that, in language, the meanings of signs are not fixed but fluctuate within a certain range. Utterances of one sign are always different from each other because of their relations with other parts. We use distributed representations to describe the differences between different utterances of the same sign. By doing this, we consider not only the differences between signs but also the differences between utterances. In our work, we focus on entities in discourse. As the narration in discourse goes, the meanings or states of entities also change. Information
carried by discourse can be encoded in the different states of entities. A memory pool is used to store all the states of entities. When a question regarding these entities comes, we can check the states of entities and give the answer based on the states of related entities. This is the entity-based memory network we proposed for reading comprehension.

The entity-centric discourse representation is also used for machine translation. It serves as a novel intermediate representation for different languages. Sentences of the source side and the target side are all represented using entities they contain. The translation is conducted at the entity level in the vector space. Since entities have references in the external world, two entity mentions (signifier) which refer to the same entity (signified) always correspond to each other. The correspondence between entity mentions is stable and persistent. It serves as the anchor in translation. The proposed model fully explore this point to build a high quality translation system.

For a given sentence, we firstly turn it into vectors of entities. Then these vectors are mapped into another vector space. Note the mapping is not arbitrary. If a vector representation $V$ represents entity $E$ in vector space $A$, then its counterpart $V' = Map(V)$ in another vector space $B$ also represents $E$. The translator is in charge of the translation of states of entities. We do not rely on it to generate valid text. We then use another model to assemble the translated parts into a new piece of discourse.

As can be seen, the novel entity-centric view of discourse brings forth new methods for dealing with practical applications. However, our work is only a preliminary study of discourse analysis. The entity structure somehow casts insights on the nature of language. But more challenges are to be met towards intelligent NLP systems. One is to develop more powerful representations for the discourse structure. Currently the entity structure which we use to analyze discourse is only capable of capturing part of the information. In particular, the relations, which are abstract and volatile, are hard to model. More sophisticated representations and models are needed to deal with relations.

Another aspect is the usage of world knowledge. Language comes from our practice in real world and reflects the knowledge obtained from real world. Con-
Considering knowledge obtained from outside sources like Freebase or Wikipedia will definitely help generate high-quality results for NLP tasks. The problem is that outside knowledge contains enormous noise which will bring negative impacts to the model if we cannot identify the information we need.

Some other problems also draw our attentions, like interpretation of neural models and generation of controllable text. We will focus on these aspects to develop more intelligent deep discourse analysis systems and leverage the obtained information to probe the mysteries of language and bring benefit to practical applications.

Work presented in this thesis leverages existing knowledge about the nature of language to design novel methods for practical applications. The exploration also helps deepen our understanding of text. But the simulation of discourse using entities is still immature due to the limitation of both tools and theories. In the future, we will further refine the entity-centric discourse analysis methods towards a deep and thorough understanding of discourse.
Bibliography


List of Publications


List of Other Publications


The contents of chapter 3 were first published in CIKM’15. Copyright©ACM. DOI:http://dx.doi.org/10.1145/2806416.2806637.

The contents of chapter 4 were first published in NAACL-HTL’15. Copyright©ACL. DOI:http://dx.doi.org/10.3115/v1/N15-1030.

The contents of chapter 5 were first published in COLING’16. Copyright©ICCL.