**JAIST Repository**
https://dspace.jaist.ac.jp/

| | |
|---|---|
| Title | A Study on Statistical Generation of a Hierarchical Structure of Topic-information for Multi-documents |
| Author(s) | NGUYEN, Viet Cuong |
| Citation | |
| Issue Date | 2011-03 |
| Type | Thesis or Dissertation |
| Text version | author |
| URL | http://hdl.handle.net/10119/12059 |
| Rights | |
| Description | Supervisor:島津　明, 情報科学研究科, 博士 |

Japan Advanced Institute of Science and Technology

# A Study on Statistical Generation of a Hierarchical Structure of Topic-information for Multi-documents

by

NGUYEN Viet Cuong

**submitted to**
**Japan Advanced Institute of Science and Technology**
**in partial fulfillment of the requirements**
**for the degree of**
**Doctor of Philosophy**

*Supervisor:* Professor Akira SHIMAZU

*School of Information Science*
*Japan Advanced Institute of Science and Technology*

March 2011

© 2011

**NGUYEN Viet Cuong**

All Rights Reserved.

*To my lovely wife,    Nguyen Thi Thuy Linh.*

*To my little boy,    Nguyen Phu Minh Duc.*

*To my parents,    Nguyen Phu Hy and Nguyen Thi Trang.*

# Abstract

Generating a hierarchical structure of topic-information (HST) for multiple documents written about the same topic is a new task in natural language processing. In a HST, topic-information is represented in a phrase and it can be seen as a title. Intuitively, a HST looks like a table-of-contents which is normally presented at the beginning of a book. It could play as a navigation tool to help readers quickly locate interesting parts. In addition, readers could look through a HST to get an overview of the topic of the document set. In this study, we propose a framework for generating a HST for multi-documents which involves three sequential tasks:

**Text segmentation** is a task of splitting a document into topically coherent segments. All documents in the set are put into a text segmentation system to get a collection of segments.

**Segment combination** is a task of merging and combining all the segments to form a hierarchical structure of segments (a tree of segments) which reflects the hierarchical structure of information.

**Title generation** is a task of generating a title for each node in the tree of segments. A title is a phrase which reflects the content of segments belonging to the node.

In the last decade, the text segmentation and title generation tasks have been received much attention from the research community. Although there are many studies investigated on various methods for these problems, the performance of available systems or published results are limited. Therefore, they are still open problems and challenges in the natural language processing field. Besides, the segment combination task is a particular task which is raised from our model. The literature related to that task is relatively sparse. Those open challenges are reasons for us to make a study on generating a HST for multi-documents. In addition, due to the dramatically improvement of computing power, people can now deal with problems which use large corpora and need high speed computation.

In this study, we aim to improve the performance of the above three tasks by using supportive knowledge in terms of semantic and topic information. The supportive knowledge which is a kind of semantic knowledge has been acquired from a large collection of texts by unsupervised learning algorithms such as word clustering and topic modeling.

The major research problems and our contributions are summarized as follows.

- First, the task of generating a HST for multi-documents is new. Therefore, we propose a framework which integrates the above three tasks in a pipeline to receive a set of documents as the input and produce a HST as the output. This framework allows us to improve the performance of tasks individually.

- Second, we focus on improving the performance of the unsupervised linear text segmentation. The current works on the task are mainly based on the assumption of lexical cohesion which consists of reiteration and collocation relations. However, they only take into account the first type of relations which can be easily recognized by observing the repetition of words. The second type of relations includes systematic and non-systematic semantic relation, which are the most complex relations to be recognized. In this study, we investigate on linguistics phenomena to find that supportive knowledge could be used to recognize these relations effectively. In addition, we also generalized current unsupervised text segmentation methods in a unique framework. The evaluation on public corpora shows the advantages of our model over the current state-of-the-art models.

- Third, the current learning models for the title generation task are still using non-semantic features about words in a text such as frequency, position, part-of-speech, syntactic function, and so on. That may be reason of the low quality of generated titles of current models. In this study, we investigate on a method to integrate semantic and topic information to the title generation learning model by using supportive knowledge. In addition, due to the lack of training data, we also investigate on using the word clustering to avoid the sparseness of data. We evaluated our proposed approach on a public dataset and get potential results.

- Finally, we investigate on the segment combination task which is raised from our framework for HST generation for multi-documents. In this study, we proposed a combination algorithm which is based on the hierarchical agglomerative clustering (HAC) method. This algorithm combines segments by the degree of topic relation between segments. The output of the algorithm is a tree which reflects the hierarchical structure of information. We also propose a heuristic algorithm to flatten the binary tree which is the output of the HAC-based algorithm to make the output look more realistic.

In summary, main contributions of this study are to propose a framework for generating a HST for multi-documents and to investigate on using supportive knowledge to improve the performance of the text segmentation and title generation tasks. The improved systems have been evaluated on the public datasets in comparison to the current state-of-the-art methods. We also did experiments on real datasets to verify the practical use of the framework.

**Keywords:** text summarization, multi-document summarization, text segmentation, title generation, supportive knowledge, topic modeling word clustering, lexical cohesion, semantic relation, semi-supervised learning, incremental perceptron.

# Acknowledgments

First and foremost I offer my sincerest gratitude to Prof. Akira Shimazu who is my supervisor at School of Information Science, JAIST, for his encouragement, guidance and support throughout my study with his patience and knowledge. You have offered much advice and insight through my work and also my life. I am really proud to be your student.

I wish to say grateful thanks to my co-advisor, Prof. Kiyoaki Shirai, for valuable discussion and comments on my presentations and my thesis.

I would like to show my gratitude to my old teacher, Prof. Ha Quang Thuy, College of Technology, Vietnam National University, Hanoi, who has taught me so much and always encouraged me during my way on research. Your recommendation is a turning point of my life.

I would like to say my special thank to Dr. Nguyen Le Minh, School of Information Science, JAIST, who is always ready to discuss with me and guide me whenever I meet tackles in research. And my colleagues in NLP Lab, I would not come to this point without your discussion and encouragement. Thank you, mates.

I wish to say sincere thanks to Prof. Ho Tu Bao and Dr. Dam Hieu Chi, School of Knowledge Science, JAIST, for your valuable advice on my work and my life. I hope that there will be more and more Vietnamese students receiving your support and recommendation to come to JAIST for advanced study.

I have received a lot of help from my "senpai" since I have started to learn informatics. Let me say special thanks to Dr. Kieu Van Hung, Dr. Trinh Dinh Thang, Dr. Trinh Dinh Vinh and Dr. Nguyen Quang Huy, Hanoi Pedagogical University No. 2, Dr. Le Quang Hieu and Dr. Phan Xuan Hieu, College of Technology, Vietnam National University, Hanoi.

I wish to send my deep acknowledgments to the Graduate Research Program (GRP) of School of Information Science, JAIST for supporting me during the past three years.

It is my lucky to become a member of Vietnamese community at JAIST. Thank you all for being with me and my family, for your warm sharing in daily life during the years. I will never forget the time with you.

Finally, my family is really the biggest motivation behind me. My lovely wife and little boy, you are the biggest gifts of my life. Thank you for coming into my life. My dear parents, your love made me today. This thesis is dedicated to you.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

## 1.1 Research Context

Nowadays, with the growth of the Internet, people are flooded with tons of information. This situation is also known as *information overload*, which is caused by a number of reasons such as highly procedure rate of new information, the ease of duplication and transmission of data, a large number of the available information channels, the contradiction and inaccuracies of information, and so on. A typical person in this Internet era normally starts a morning with checking e-mails, reading online newspapers, surfing some blogs, etc. to update information. They do such time-consuming tasks because of their jobs or just their habits. To help people save time, many Internet-based companies run news aggregation websites which collect news articles, blogs, podcasts, etc. and put them into a single location for easy accessing. *News aggregators* may collect news articles manually as Drugde Report[1] and Huffington Post[2], or entirely automatic as Google News[3] and Techmeme[4]. Then, the articles are grouped by categories such as politics, entertainment, science, etc. or by events such as "Hanvon to introduce a color e-ink reader," "Japan encourages encourage Vietnam to buy Shinkansen technology." Although news aggregators help people so much in accessing articles written about the same topic, the number of pages to be read is still very large. They also contain much duplicate information or even contradictory and inaccurate information. Consequently, it is very difficult for a person to read all the related articles to get an overview of interesting parts of the topic. A solution is to read a summary of articles, which presents important information in a concise form to dramatically reduce the reading time. Multi-document summarization is a field in natural language processing that has been invented to deal with that problem.

Recently, multi-document summarization has received much attention from the research community. The NIST[5] has conducted a series of workshops and conferences such as DUC 2001-2007 and TAC 2008-2010 for challenging researchers on text summarization and multi-document summarization for years. There are also some online multi-document

[1]`http://www.drudgereport.com/`
[2]`http://www.huffingtonpost.com/`
[3]`http://news.google.com/`
[4]`http://www.techmeme.com/`
[5]National Institute of Standards and Technology, USA

summarization systems built by research groups such as NewsInEssence [85] of CLAIR[6] group at University of Michigan and Newsblaster [64] of NLP[7] group at Columbia University.

Multi-document summarization is an automatic process that aims to extract information from multiple texts written about the same topic and put them into a concise and comprehensive report. The resulting report allows individual users, or even professional information consumers, to quickly get an overview of information contained in a large cluster of documents. In such a way, multi-document summarization is a natural evolutionary step of news aggregators. However, the output of a typical summarization system is normally a text which is constructed by sentences, which are, in turn, extracted or generated from the set of documents. With this type of representation, people still have to read through the summary and organize information in a structure by themselves. This type of summary even contains sentences with different writing styles. This is especially much more difficult for non-native speakers. In the real life, there is a special type of summary placed in the beginning of every book, which is a table-of-contents.

A table-of-contents is a *hierarchical structure of topic-information* (HST) that can be used as a navigation tool to locate interested sections or get an overview of the contents of a book. A HST is usually used in a long text and can be built from the readily hierarchical structure of contents such as parts, chapters, sections, and so on. Generating a HST for a long text, such as book, has been firstly introduced in [4] with an unsupervised approach based on lexical chain assumption. In [17], the authors proposed a statistical model for generating a table-of-contents for a book which has a readily hierarchical structure of contents.

In this study, we aim to develop a framework for generating a hierarchical structure of topic-information for multi-documents, in which the topic-information is represented in form of a phrase or a title. This model could not be easily extended from the previous works [4, 17] for single document because it must be deal with a number of problems of multi-documents such as redundancies, differences, and conflicts. It is also different from previous multi-document summarization methods [49] since it aims to generate titles, which are very short texts, to represent the topic-information of the document set. It also has to discover the hierarchical structure of contents inside the document set.

In the scope of this study, we propose a three-step framework for generating a HST for multi-documents. Figure 1.1 shows the framework with three major tasks: *text segmentation*, *segment combination*, and *title generation*. Firstly, every document in a cluster has been split into segments. Secondly, all segments have been combined into a tree which represents the hierarchical structure of information. Finally, a title has been generated for each segment. Those titles in combination with the tree of segments form a HST for multi-documents.

[6]`http://clair.si.umich.edu/`
[7]`http://www1.cs.columbia.edu/nlp/`

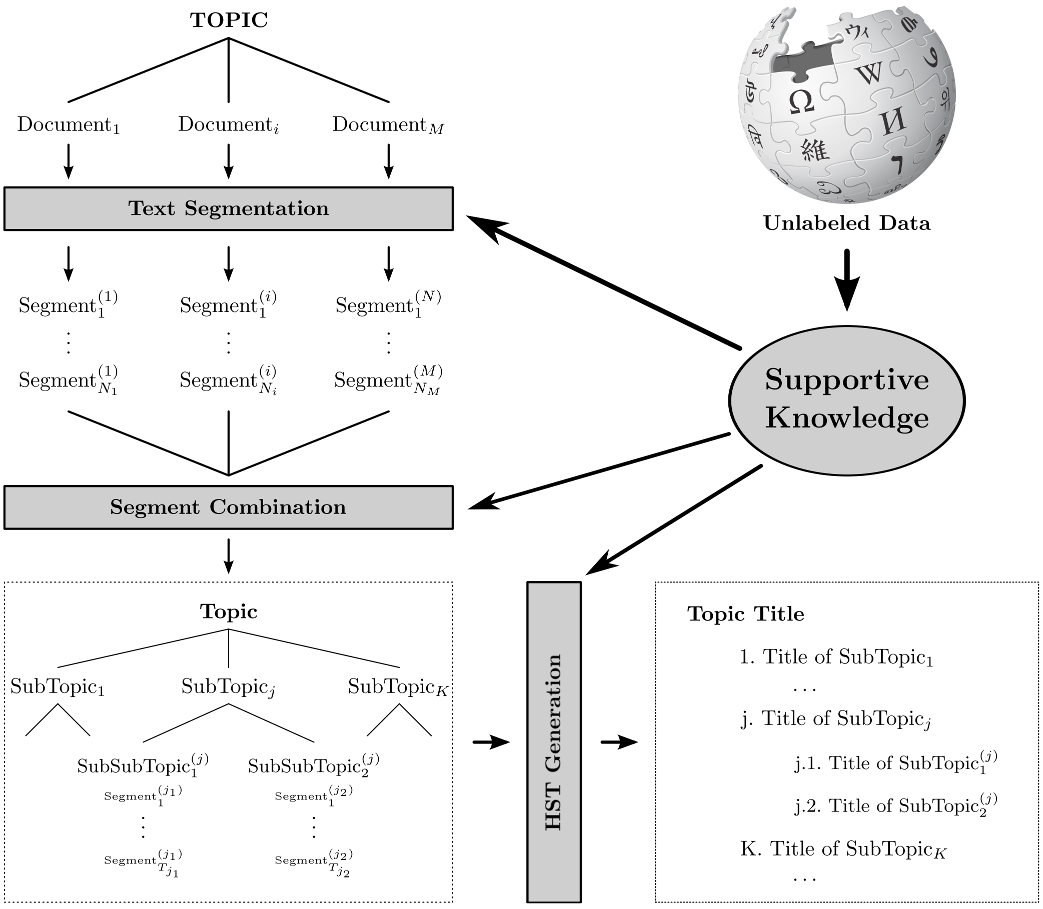


Figure 1.1: A framework for generating a HST for multi-documents.

## 1.2 Motivations and Contributions

The main goal of this research is to build a system that generates a high quality HST for multi-documents with the minimum human effort in constructing linguistic knowledge. For this reason, we focus on using *supportive knowledge* resources acquired from unlabeled data, which is easily crawled from the Internet. These resources are used in our model to improve the quality of the resulting HST.

The definitions of tasks in this study with our contributions are described as follows.

### 1.2.1 Supportive Knowledge

In this study, we define supportive knowledge as a kind of semantic knowledge used to support statistical models in all three tasks: text segmentation, segment combination, and title generation. An important point of supportive knowledge used in this study is that it has been acquired from a collection of texts by an entirely automatic process. The collection of texts is also freely available on the Internet. In this study, we investigate two methods to acquired supportive knowledge, which are word clustering and topic modeling. The both two methods create clusters of words based on their co-occurrence information (collocation), in which the former produce hard clusters and the latter produce soft clusters. In the other hand, we also investigate methods to exploit the structure of word clustering and topic modeling to compute the semantic relation between two linguistic units such as sentence vs. sentence, sentence vs. text, and word vs. word.

Some works which employ supportive knowledge acquired in the scope of this research are reported in [74, 73, 72, 75, 76].

### 1.2.2 Text Segmentation

Text segmentation is a process of splitting a document or a continuous stream of text into topically coherent segments. Text segmentation methods can be divided into two categories by the structure of output that is linear segmentation [43, 87, 8, 22, 96, 59] and hierarchical segmentation [103, 91, 33], or by the algorithms that are unsupervised segmentation or supervised segmentation. The main advantage of unsupervised approach is that it does not require labeled data and is domain independent.

In this study, we focus on unsupervised-linear text segmentation methods with the assumption of lexical cohesion [41]. Halliday and Hasan [41] defined two categories of lexical cohesion: reiteration and collocation. The current approaches in lexical cohesion-based text segmentation only focus on the first category of lexical cohesion, reiteration [43, 87, 22, 59], with the repetition of words can play as the indicator of the topic coherence in a segment and the topic incoherent between segments.

To cope with the second category of lexical cohesion, we propose a method to recognize collocation relations in order to improve the performance of the text segmentation system. Specifically, the supportive knowledge is used in our model to capture systematic semantic relation and non-systematic semantic relation, which are two relationships in collocation.

Some results of our work on this task are reported in [75].

### 1.2.3 Segment Combination

Segment combination is an immediate step that is raised in our model for generating a HST. The main purpose of this step is to build a composite tree of segments which are produced by the text segmentation step. This hierarchical structure will be the input for the next step, title generation. The resulting tree should contain segments with the similar content in the same sub-tree. Furthermore, the content of an inside node should be more general than the content of belonging leaf nodes.

In this study, we propose an algorithm for building such a tree based on the hierarchical agglomerative clustering (HAC) method. We also propose the way of using supportive knowledge in that algorithm based on the idea that the distribution of topic of the cluster at the higher levels should be more *general* than the distribution of topic of the cluster at the lower levels. Specifically, a cluster at the higher level should have higher entropy in comparison to the lower one.

### 1.2.4 Title Generation

Title is a very short text that provides a compact representation of the content of the document. Therefore, it helps people quickly capture the main idea of a document without spending time on the detail. Title generation is a very complex task. A title generation algorithm should not only choose the appropriate words that reflects the main content of the document, but also has right order of words for the readability.

In the past decade, although there is a large number of works on text summarization, there are still a small number of researches on title generation, headline generation, or very short text summarization [47, 99, 100]. The current approaches can be divided into three categories based on the method of generating title: key phrase extraction, sentence compression, and statistical generation.

In this study, we follow the statistical generation approach. The principle of statistical approach is to first learn the correlation between the words in titles and the words in the corresponding documents from a given training corpus, and then apply the learned correlations to generate titles for new documents [47]. However, the current researches on title generation still use only lexical features from the document or a little features from the syntactic tree. We propose a further step on incorporating semantic information into the title generation learning. Our approach is based on the idea that *a good title should have a topic relation to the text.* This idea comes from the characteristics of a title, which is mentioned above. Therefore, the supportive knowledge is, again, useful in this task. In our model, topic modeling is used to take into account the overlap of topic information between document and title, and word clustering is used to deal with the sparseness problem.

Some results of our work on this task are reported in [74, 73, 76].

One of the advantages of our HST generation model is that it can be applied into multiple domains and multiple languages. The reason is that our model is trained without domain-specific features. In addition, it is also not depend on language-specific knowledge resources such as semantic nets.

## 1.3 Thesis Structure

The remainder of this thesis is organized as follows: Chapter 2 briefly presents the background knowledge that is useful for understanding tasks in our HST model. The advantages and disadvantages of current approaches on those tasks are also discussed. In the main portion of the thesis, Chapters 3 through 6, we investigate the tasks in HST generation in detail with our contributions. Our proposed models are evaluated on the public datasets in comparison to current state-of-the-art models. The last chapter give some conclusions with future works.

The road map of this thesis is based on steps in our proposed model in Figure 1.1. The contents of the remaining chapters can be outlined as follows.

**Chapter 2** presents the main points of the text segmentation and title generation tasks.
We first survey the current approaches on the text segmentation task. We also

present a unified framework for the lexical-based unsupervised text segmentation algorithms. Based on that framework, one can make changes in some parts of the framework to improve the overall performance of a text segmentation system. We finish the discussion on the text segmentation task with the presentation of evaluation measures, which are much different from the traditional measures such as precision, recall, and F-score. We then briefly present the current approaches on title generation task, including key phrase extraction, sentence compression, and statistical generation.

**Chapter 3** presents one of the main points of this thesis-supportive knowledge. We first introduce the supportive knowledge and the previous works. We then focus on two models that are used to derive supportive knowledge from unlabeled data, which are word clustering and topic modeling. We also present our work on collecting data and deriving the supportive knowledge from a free and large collection of unlabeled data, WIKI dataset. We finish the chapter with some experiments on the collected data with some discussion.

**Chapter 4** presents our work on improving the performance of the text segmentation with systematic and non-systematic semantic relation. We first summarize the limitation of current approaches on the lexical-based unsupervised text segmentation algorithms. We then introduce the other parts of lexical cohesion and discuss the impact of them on text segmentation task under the general framework presented in Chapter 2. Next, we propose a method to exploit supportive knowledge in text segmentation task to recognize the collocation relationship. We finish this chapter with some experiments on the widely used dataset for this task. We also compare the experimental results of our model with available text segmentation systems.

**Chapter 5** presents our work on the HST generation task. We first present the supervised learning model for this task, which is mainly based on the statistical generation models for title generation. The title generation process is modeled with an incremental perceptron model. We then follow the semi-supervised approach to incorporate the supportive knowledge into the supervised learning model to capture the topic relation between a title and the corresponding segment of text. The features used in title generation are also deeply analyzed and discussed. We last do some experiments on the public dataset to show the advantage of our approach in comparison to the current state-of-the-art model.

**Chapter 6** combines our works in Chapters 3, 4, and 5 to generate a HST for multi-document written about the same topic. We first discuss some differences on applying text segmentation to a set of relevant documents with some advantages and disadvantages. We then propose a clustering-based model for combining segments into a composite tree that reflects the hierarchical structure of information inside a set of documents. We also discuss some important points to apply the HST generation model to the multi-document case. We finish this chapter with some experiments on real data.

**Chapter 7** firstly summarizes main points of this thesis with our main contributions as well as the remaining problems. Next, we present some extendable parts of this study for the future research directions.

In addition, we also provide two appendices. Appendix A gives definitions and examples of cohesion in English. Appendix B briefly introduces tools and datasets used in this study.

# Chapter 2

# Background

In this chapter, we present background knowledge about the tasks in our HST generation model. First, we give an overview of the text segmentation task. Second, the current studies on the title generation task are surveyed. Last, we briefly introduce some machine learning methods used in this study.

## 2.1 Text Segmentation

Text segmentation is one of the fundamental problems in natural language processing. It is a process of splitting a document or a continuous stream of text into topically coherent segments. Text segmentation methods can be divided into two categories by the structure of output that is linear segmentation [6, 22, 34, 42, 46, 59, 67, 87, 96] and hierarchical segmentation [79], or by the algorithms that are unsupervised segmentation or supervised segmentation. In this study, we focus on the unsupervised-linear text segmentation method. The main advantage of unsupervised approach is that it does not require labeled data and is domain independent.

Linear text segmentation has many important applications in natural language processing. In *information retrieval*, a system normally search and send documents that contains what the user needs. However, with a long document, a natural user's demand is that the information retrieval system can point out which parts are relevant to the user's query. In addition, in the text streams of news broadcast or the automatic speech recognition transcripts, the boundaries between documents are not explicitly marked [59]. Human can easily recognize those boundaries, but for a small number of documents. Therefore, automatic text segmentaion is a critical task in such systems for accessing information. In *text summarization*, a document often discuss multiple sub-topics that are relevant to the main topic. With the discovered topical structure of the document, a summarization system can produce a summary that covers almost important parts [6].

Almost unsupervised text segmentation methods are based on the assumption of cohesion [41], which is a device for making connection between parts of the text. Cohesion is achieved through the use of reference, substitution, ellipsis, conjunction, and lexical cohesion. The most frequent type is lexical cohesion, which is created by using semantically related words. Halliday and Hasan in [41] classified lexical cohesion into two categories:

Table 2.1: Five types of relationships in lexical cohesion

| No. | Type of relation | Example |
|---|---|---|
| 1 | Reiteration with identity of reference | Mary bit into a *peach.* Unfortunately, the *peach* wasn't ripe. |
| 2 | Reiteration without identity of reference | Mary ate some *peaches.* She likes *peaches* very much. |
| 3 | Reiteration by means of superordinate | Mary ate a *peach.* She likes *fruit.* |
| 4 | Systematic semantic relation | Mary likes *green* apples. She does not like *red* ones. |
| 5 | Non-systematic semantic relation | Mary spent three hours in the *garden* yesterday. She was *digging* potatoes. |

reiteration and collocation. Reiteration includes word repetition, synonym, and superordinate. Collocation includes relations between words that tend to co-occur in the same contexts which are the systematic and the non-systematic semantic relations.

In the next section, we briefly introduce linguistic foundation about lexical cohesion [41, 87] and show that how it acts an important role in text segmentation. We then discuss both unsupervised and supervised approach in text segmentation. We also summarize the current approaches to unsupervised-linear text segmentation in a general framework. Based on that framework, one can easily improve the performance of a text segmentation system. Finally, we describe measures that are used to evaluate text segmentation algorithms.

### 2.1.1 Lexical Cohesion

Almost unsupervised text segmentation algorithms are based on the assumption that the lexical repetition indicates topic continuity, while changes in lexical distribution indicates topic changes [6, 22, 34, 42, 46, 59, 87].

Halliday and Hasan [41] was the first that has a deep investigation on *cohesion* in English. In [41], they present five types of cohesion that form the texture, which is the main materials to make a sequence of sentences become a text. Those are reference, substitution, ellipsis, conjunction, and lexical cohesion. The first four types are at the syntactic level, and the last one is at the word surface level. Based on the appearance of cohesion, one can determine the topical structure of a text.

Morris and Hirst [68] were the first to apply lexical cohesion for text segmentation. Based on the reiteration and collocation relationships in [41], they divided lexical cohesion into five types of relationships that are presented in Table 2.1. The reiteration includes not only identity of reference or word repetition, but also the use of synonym or superordinate.

The collocation includes semantic relationships between words that often co-occur. They can be further divided into two categories of relationship: systematic semantic and non-systematic semantic. The definitions of five types of relationships with some examples are presented in Appendix A.

Figure 2.1 illustrates the concept of lexical cohesion on two consecutive segments of text extracted from the article “Stargazers”, which is an well-known example in literature on the text segmentation task [42, 43, 96, 50]. This example was manually segmented and entitled by Hearst [42]. The first segment discusses “The moon’s chemical composition”, and the segment section discusses “How early earth-moon proximity shaped the moon”. The words that repeated in each segment are superscripted with a number indicating their group.

Lexical cohesion in two segments in Figure 2.1 can be observed through the repetition of key topical words at the surface level of sentences. For example, the words “material” and “form” is repeated through the sentences of the first segment and do not appear in the second segment. In addition, some words such as “metals”, “iron”, “silicate”, “mineral”, “element” which have different meanings but the same topic in this context are evidence for non-systematic semantic relation. Those words relate to the topic of “chemical composition”. Likewise, in the second segment, the repetition of words “measurement”, “surface”, “position”, “orbit”, “mass”... relates to the topic of “earth-moon proximity shaped the moon”. In the other hand, the appearance of words “earth” and “moon” in almost all of the sentences in both segments indicate that the topic of two segments might be the relationship of earth and moon. In general, if the topics of two segments are sufficiently different, it should be expected that the associated key topical words will be different as well.

This repetition property can be exploited for recognizing the topic shift within a text. Specifically, spans of text that have similar lexical distribution tend to be in the same topical segment. Therefore, the boundaries should be chosen at locations of prominent change in lexical distribution. In addition to the word repetition, synonyms, hyponyms and word collocations are also the notation of the continuity of a topic. In our example, the semantically related words “iron”, “silicate”, “material” have collocation relationship in terms of co-occurence. Those words are normally co-occur in the same document and semantically related. Thus, they can form a relation between two text spans. Despite being patently obvious, the lexical cohesion is very powerful because its degree can be quantified through simple word matching.

Besides lexical cohesion, Halliday and Hasan establish that the presence of certain semantic devices in the text can crystallize the latent thematic structure. Conjunctions such as “Moreover” in the above text, point to associations between adjoining clauses or sentences. Referential links between anaphors and their antecedents also preserve continuity of the spanned text fragments, because of the persistence of the underlying object. So, in the first segment, “that object” is referring to the previously mentioned idea. Finally, substitution and ellipsis are also quite common devices that elicit cohesion. These correspond to cases where certain word phrases are implicitly acknowledged to have been either replaced by simpler referring expressions or removed altogether.

In text segmentation task, all the semantic devices in cohesion can be used to eliminate or identify potential segment boundaries. For example, lexical items and cue words that

Relative to its own size, no other planet has such a big moon$^1$ as earth$^2$. Moreover, studies of moon$^1$ rock brought by Apollo astronauts suggest that the moon$^1$ formed$^3$ from a large object that had already cooled from a molten state during which heavy metals$^5$ such as iron$^5$ had gathered in the core leaving lighter, silicate$^5$ materials$^4$ to form$^3$ a crust. That object was the earth$^2$ itself. Chemical analysis of lunar samples and accurate dating confirms that the moon$^1$ formed$^3$ very soon after the earth$^2$. But the only satisfactory theory to explain the origin of the moon$^1$ as a separate body in space says that a massive body such as an asteroid collided with the earth$^2$ and ejected a chunk of material$^4$ which cooled to form$^3$ the moon$^1$. Minerals$^5$ in lunar samples are remarkably similar to materials$^4$ comprising the earth$^2$'s outer mantle and crust.
The moon$^1$ is generally made up of much lighter materials$^4$ than the earth$^2$ and the other terrestrial planets (Mercury, Mars and Venus). It also has much less iron$^5$ and other dense elements$^5$ than are typical in a planet like earth$^2$ that emerged from a condensing cloud of gas$^5$ when the sun formed$^3$ as a star. The difference has always confused astronomers, and the new evidence helps to explain this.

---

Following careful measurements$^6$ of the moon$^1$'s position$^8$, scientists are now sure it used to be much closer$^9$ to the earth$^2$ and that it is slowly drifting away. To examine this, scientists use special reflectors left on the lunar surface$^7$. They can measure$^6$ the distance$^9$ between the earth$^2$ and the moon$^1$ to an accuracy of 5cm. This is done by bouncing laser beams off the reflectors and measuring distance$^9$ by calculating the time taken for the beam to reach the moon$^1$ and return. The information also helps establish the earth$^2$ and the moon$^1$'s exact mass$^8$, data vital for the computer simulation of the moon$^1$'s orbit$^8$.
The conclusions show the moon$^1$ was originally only 20,000 km away, against 384,000 km today. This is confirmed by traces on old ocean shores where tides of 300m, caused by a much greater pull from the moon$^1$, were not uncommon . The pull was so great that the moon$^1$ would have had to be much closer$^9$ to exert that effect on the earth$^2$'s oceans.
The effect of the earth$^2$ on the moon$^1$ when it was much closer$^9$ is marked by the light and dark patches across the latter's surface$^7$. The dark smudges are dried lakes of lava that, more than 2-billion years ago, oozed forth across the nearside of the lunar surface$^7$ as the earth$^2$ exerted its influence on the moon$^1$'s interior. Spacecraft that photographed the hidden face of the moon$^1$ reveal an absence of these dry lava lakes on the far$^9$ side. Indirect measurements$^6$ of the moon$^1$'s interior show the molten layer under the crust to have a distinct pear shape, with the greatest mass$^8$ pulled off-centre in the direction of the earth$^2$. These are further$^9$ indications that it was once very close$^9$ to our planet.

Figure 2.1: An example of lexical cohesion on the article "Stargazers".

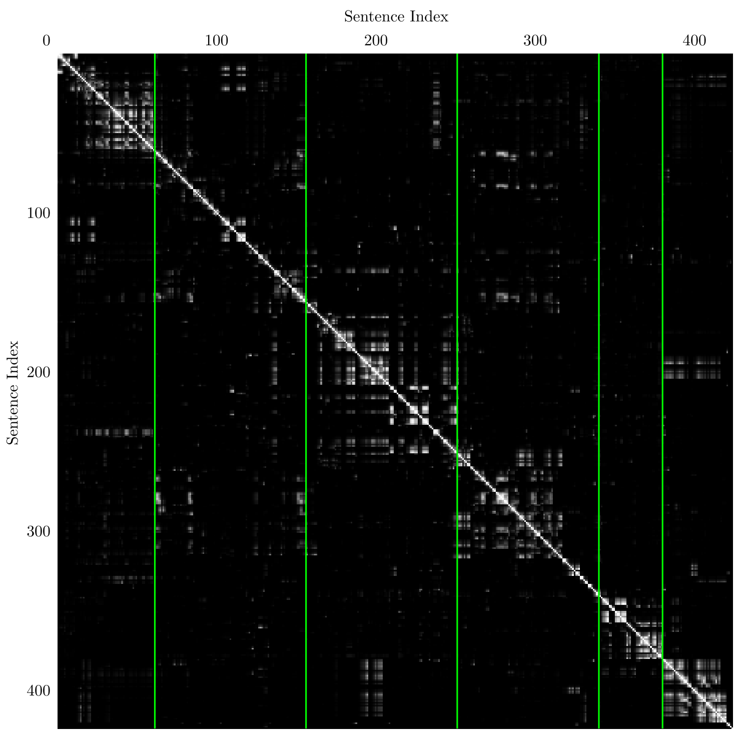


Figure 2.2: DotPlot for a transcribed AI lecture with vertical lines indicating true segment boundaries.

usually tend to signal references, substitutions, and conjunctions can be readily identified. These trigger words are usually used in supervised segmentation systems in form of lexical features. In [87], the author observes that anaphoric links tend to occur much more frequently within segments than across different segments and registers the presence of anaphoric links as a feature in the segmentation system. This analysis is consistent with the linguistic function of reference in eliciting cohesion.

### Empirical Basis of Lexical Cohesion

Church [24] used a simple graphical representation to model the lexical distribution in text. He ploted the cosine similarity scores between every pair of vector representation of sentences in the text. The intensity of a point $(i, j)$ on the plot indicates the degree to which the $i$-th sentence is similar to the $j$-th sentence. He called it a *DotPlot*.

Figure 2.2 is a DotPlot for a transcribed AI lecture. The vertical green lines indicate the true segment boundaries. This similarity plot reveals a block structure where true boundaries delimit blocks of text with high inter-sentential similarity. Sentences found in different blocks, on the other hand, tend to exhibit low similarity.

The relation between every pair of sentences in the text can be also represented as a graph, in which a vertex represents a sentence or a block of text, and an edge represents the degree of relation between the two associated sentences. This graph is actually the underlining representation of the DotPlot. However, it makes easier to apply graph algorithms on the relationship network of sentences.

Under multiple domains in both written and spoken genres of language, this repre-

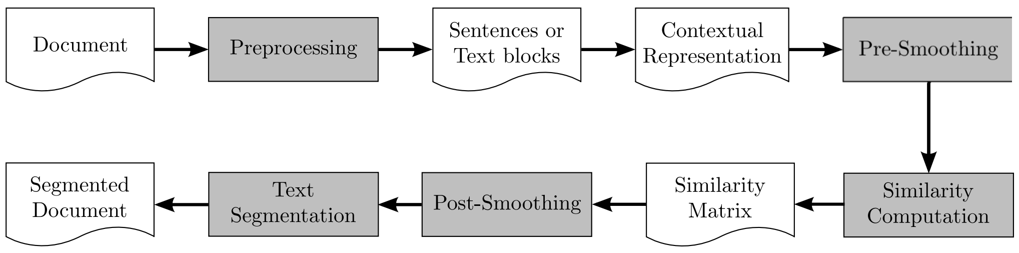


Figure 2.3: The general framework of the similarity-based text segmentation algorithm

sentation consistently bears out the claim that repetition of content words is a strong indicator of thematic cohesion, while changes in the lexical distributions usually signal topic transitions. In fact, this representation serves as a basis for many unsupervised algorithms, including the recent approach in [59] and the approach proposed in this thesis.

### 2.1.2 Unsupervised Approaches

Algorithms for unsupervised text segmentation could be divided into two categories: lexical cohesion-based [22, 42, 46, 59, 87] and generative-based [96, 34]. The lexical cohesion-based approaches could be, in turn, divided into lexical chain-based and similarity-based. In deed, the different between two sub-categories is minor because they are also based on the principle of the lexical chain [68]. Figure 2.3 shows the general framework for similarity-based text segmentation.

This process could be interpreted as follows. First, a document has been split into sentences or fixed-size blocks of texts. Then, the contextual representation, which is normally occurrence matrix, is built based on a vocabulary, in which one dimension is for sentences, and another dimension is for words in the vocabulary. To remove some gaps that are created by short sentences or sentences containing common words, some smoothing technique might be applied on the occurrence matrix. The next step is creating a similarity-distance matrix between all pairs of sentences. This matrix is normally seen as a gray-scale image, which is called DotPlot [87]. Thus, the text segmentation problem can be seen as a special case of the image segmentation problem or the graph partitioning problem. As common in image processing, some smoothing techniques may be applied to enhance density of some area and reduce noise. Last, a segmentation algorithm has been applied on the similarity matrix or DotPlot image to find the boundaries of segments in the given document. Although the graph partitioning problem is NP-complete, we can easily create a dynamic programming algorithm based on the linear characteristics of the text segmentation problem.

Previous approaches are normally different in the contextual representation, the similarity matrix computation, the smoothing technique, and the segmentation algorithm. The detail of such parts are presented as follows.

#### Contextual Representation

The contextual representation is normally the occurrence matrix or lexical weighting matrix, in which a cell contains a number that represent the frequency of a word in a sentence [42, 86]. In [22], he compute TF-IDF score for words in a text by split a that text into equal chunks, where a chunk is treated as a document. The TF is the term frequency of a word in its container, and the IDF is the inverse chunk frequency of the same word over whole text. The container here may be a sentence or a block of text with fixed size. This technique is then also employed in [59]. In [23], the authors make a further step on representing the lexical weight. They refine the lexical weighting matrix by incorporating Latent Semantic Analysis (LSA) [29].

### Similarity Matrix Computation

Based on the contextual representation, a simiarity-distance matrix has been computed. The similarity is measured in terms of cosine similarity of two adjacent blocks, $s_i = (w_{i_1} w_{i_2} \dots w_{i_n})$ and $s_j = (w_{j_1} w_{j_2} \dots w_{j_n})$, where cosine similarity, $\text{sim}(s_i, s_j)$, is defined as

$$\text{sim}(s_i, s_j) = \frac{s_i \cdot s_j}{||s_i|| \times ||s_j||} \tag{2.1}$$

where, $s_i \cdot s_j$ is the dot product of two vectors and $||x||$ is the $L_2$ norm of vector $x$.

In [59], they use an exponential version of similarity to accentuate differences between low and high lexical similarities $e^{\text{sim}} s_i, s_j$.

Most unsupervised text segmentation algorithms are based on the assumption that spans of text with homogeneous lexical distributions should correspond to topically coherent segments. Therefore, the homogeneity is typically computed by analyzing the similarity in the distribution of words within a segment. The approaches that maximize self-similarity within a segment include [22, 87, 46]. Other approaches determine segment boundaries by locating sharp changes in similarity of adjacent blocks of text [43, 87]. An ideal algorithm should take into account both objectives in determine segment boundaries.

### Smoothing Technique

Smoothing techniques are applied before and after the computation of similarity matrix. In our generalized framework, we called them pre-smoothing and post-smoothing, respectively.

The pre-smoothing technique is applied on the contextual representation. It is used to reduce the gaps between adjancent block of texts in case of short sentences containing too many common words. In [42, 86, 22, 23], they compute the lexical weights on a range of adjacent sentences. In [59], they employ exponentially weighted moving average (EWMA) to update the vector representation of sentences based.

The post-smoothing technique, on the other hand, is applied on the similarity-distance matrix. As mentioned above, the simiarity-distance matrix can be viewed as a weighted graph or a gray-scale image DotPlot. Thereby, one can employ smoothing techniques from image processing field. The main purpose is to reduce noise in homogeneous regions, make homogeneous regions more homogeneous, and sharpen the boundaries between homogeneous regions. For example, a rank filtering with window size $11 \times 11$ is used in [22, 23], or the anisotropic diffusion technique is employed in [46].

#### Segmentation Algorithm

The most important part of the text segmentation task is decoding algorithm or segmentation algorithm. Currently, there are two classes of decoding algorithm in this framework which are the greedy approximation and the exact inference. The first class includes the top-down clustering based algorithm proposed by Reynar [86, 87] and later used by Choi [22]. The second class is also the most popular, which finds the exact solution via a dynamic programming algorithmn [23, 46, 59].

### 2.1.3 Supervised Approaches

In the scope of this thesis, we focus on unsupervised, similarity-based models for text segmentation. However, we will briefly describe some supervised approach. These methods usually require large amounts of in-domain training, and are sensitive to noise, speech recognition errors, and data sparsity. The supervised methods for segmentation typically fall into one of the two classes, namely binary classification or sequential models.

#### Classification and Sequential Models

Under the classification framework, each candidate boundary location in the text is evaluated independently by the model, and then the top scoring candidate boundaries are selected. Some of the approaches applied to text segmentation in this class of learning algorithms in the past include Decision Trees [80], Maximum Entropy [8], Support Vector Machines [52], and Boosting [93]. The strength of these models lies in their ability to encode arbitrary local contextual features. However, the fact that hypotheses are evaluated independently detracts from their effectiveness, since segment boundaries are inter-dependent. For example, these types of models will not be able to capture the fact that very short segments should be unlikely.

Sequential models, as the name implies, model sequences of decisions. [84, 69, 90, 12] model text streams with Hidden Markov Models over word sequences, with HMM states corresponding to boundary and non-boundary states delimiting segments. [30] employed Dynamic Bayesian Networks for structured multi-party meeting segmentation. These approaches typically require a lot of training data, and they are applied to highly structured domains.

#### Features

The effectiveness of supervised segmentation models often hinges on choosing a suitable feature representation. In the written language domain, lexical cohesion and linguistically motivated features are used. Cohesion features capture the underlying word distributions, indicating whether segments are lexically cohesive. [8] encode the log likelihood of a context-sensitive and context-independent language model as a feature in their model. [37] incorporate cosine similarity scores between blocks of text. The linguistic features may register the presence of referential noun phrases, which indicate topic continuity or cue words, which usually signal topic changes. In spoken language segmentation, additional prosodic, acoustic, and discourse features such as speaker activity, speaker overlaps, and pause duration have been used to improve segmentation quality [90].

## 2.1.4 Evaluation

It is generally to evaluate a text segmentation by running the algorithm on a test set in which boundaries have been labeled by humans and then comparing the automatic and human boundary labels using the $\mathrm{P}_k$ [8] or WindowDiff [82].

We generally do not use precision, recall, and F-measure for evaluating segmentation because they are not sensitive to near misses. If a segmentation algorithm is off by one sentence in assigning a boundary, standard F-measure gives it as bad a score as an algorithm that assigned boundaries nowhere near the correct locations. Both $\mathrm{P}_k$ and WindowDiff assign partial credit. WindowDiff is a variant of $\mathrm{P}_k$.

$\mathrm{P}_k$ and WindowDiff compares a sentence (human-labeled) segmentation, or reference segmentation, with a hypothesis segmentation by sliding a probe, a moving window of length $k$, across the hypothesis segmentation. At each position in the hypothesis string, we compare the number of *reference* boundaries $r_i$ that fall within the probe to the number of *hypothesized* boundaries $h_i$ that fall within the probe. WindowDiff algorithm penalizes any hypothesis for which $r_i \neq h_i$, that is, for which $\delta(r_i - h_i) = 1$. Meanwhile, $\mathrm{P}_k$ algorithm penalizes if $\delta(r_i) \neq \delta(h_i)$. $\delta(x)$ is Dirac delta function that has the value zero except at $x = 0$. The window size $k$ is set as half the average segment in the reference string. Figure 2.4 shows a schematic of the computation.

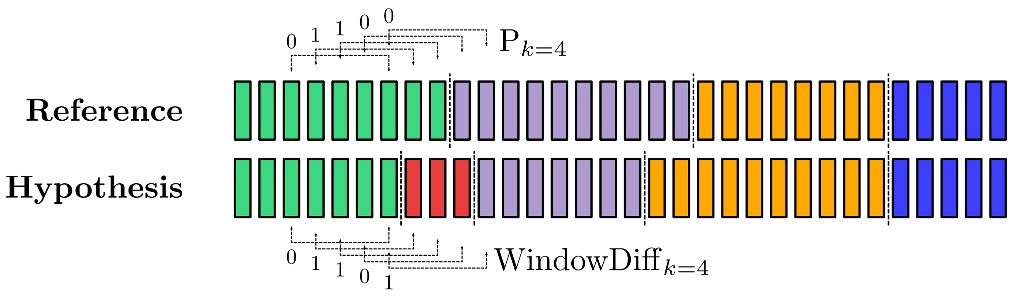


Figure 2.4: The illustration of $P_k$ and WindowDiff evaluation.

More formally, if $b(i,j)$ is the number of boundaries between positions $i$ and $j$ in a text, and $N$ is the number of sentences in the text, then

$$\mathrm{P}_k(ref, hyp) = \frac{1}{N-k}\sum_{i=1}^{N-k} \delta(\delta(b_{ref}(i,i+k)) - \delta(b_{hyp}(i,i+k))) \quad (2.2)$$

$$\mathrm{WindowDiff}(ref, hyp) = \frac{1}{N-k}\sum_{i=1}^{N-k} \delta(b_{ref}(i,i+k) - b_{hyp}(i,i+k)) \quad (2.3)$$

In [82], one of the problems of $\mathrm{P}_k$ they identify is that with greater variation in segment length, the measure becomes more lenient. The primary reason for this is that a penalty is registered only if the reference and hypothesis differ in their assignment of the sentence pair to the same segment or to two different segments. This approach will not identify errors where both the reference and the hypothesis assign sentences to different segments, yet in one segmentation there are more intervening segments than in the other. WindowDiff has been proposed to solve this problem.

## 2.2 Title Generation

*Title* is a general or descriptive heading for a section of a written work[1].

In this study, we view a title as a very short text that provides a compact representation of the content of document and therefore helps people quickly capture the main idea of a document without spending time on the details. Title creation is a complex task even for human: One has to understand what the document is about, one has to know what is characteristic of this document with respect to other documents, one has to know how a good title sounds to catch attention and how to distill the essence of the document into a title of just a few words.

Automatic title generation is also a complex task which not only requires finding the title words that reflects the document content, but also demands ordering the selected title words into a human readable sequence. Therefore, it involves in both nature language understanding and nature language synthesis, which distinguishes title generation from other seemingly similar tasks such as key phrase extraction or automatic text summarization where the main concern of tasks is identify important information units from documents [60].

In the past decade, although there is a large number of works on text summarization, there is still a small number of researches on title generation, headline generation, or very short text summarization [101, 47, 99]. We can divide the approaches into three categories based on the method of generating title: *key phrase extraction*, *sentence compression*, and *statistical generation*.

### 2.2.1 Key Phrase Extraction

This approach normally selects the key phrase from a list of noun phrases in the document to form a title [4]. The methods used to rank the extracted phrases are employed from the popular keywords extraction techniques [102].

The title extracted by this approach is normally good if the document is short and the content only concern one or two objects. That is also the disadvantage of this approach. It cannot make a title, wherein there are interactions between two or more objects.

[1]Definition in WordNet 3.0 ©2006 by Princeton University. `http://wordnet.princeton.edu`

**Lead sentence:** *The U.S. space shuttle Discovery returned home this morning after astronauts successfully ended their 10-day Hubble Space telescope service mission.*

**Step 1** Choose leftmost S (declarative clause) of syntactic tree and remove all determiners, time expressions and low content units such as quantifiers (e.g. *each*, *many*, *some*), possessive pronouns (e.g. *their*, *our*, *her*) and deictics (e.g. *this*, *these*, *those*).

```
(S  (NP  (NP  The U.S. shuttle)
         Discovery)
    (VP  returned
        (NP  home)
        (NP  this morning))
    (SBAR  after
          (S  (NP  astronauts)
              (VP  (ADVP  successfully)
                   ended
                   (NP  their 10-day Hubble Space telescope
                        service mission)))))
```

**Step 2** The next step iteratively removes constituents until the desired length is reached. In this example, the algorithm will remove the trailing SBAR (subordinate clause).

```
(S  (NP  (NP  U.S. space shuttle)
         Discovery)
    (VP  returned
         (NP  home))
    (SBAR  after
           (S  (NP  astronauts)
               (VP  (ADVP  successfully)
                    ended
                    (NP  their 10-day Hubble Space telescope
                         service mission)))))
```

**Step 3** Convert the tree to the string.

```
(S  (NP  (NP  U.S. space shuttle)
         Discovery)
    (VP  returned
         (NP  home)))
```

**Output:** *U.S. space shuttle Discovery returned home.*

Figure 2.5: An example of sentence compression approach

### 2.2.2 Sentence Compression

Dorr et al. [31] stated that when human subjects were asked to write titles by selecting words in order of occurrence in the source text, 86.8% of these headline words occurred in the first sentence of the news story. Based on this result, they concluded that compressing the lead sentence was sufficient when generating titles for news stories. Consequently, their DUC 2003 system HedgeTrimmer used linguistically-motivated heuristics to remove constituents that could be eliminated from a parse tree representation of the lead sentence without affecting the factual correctness or grammaticality of the sentence. These linguistically-motivated trimming rules [31, 106] iteratively remove constituents until a desired sentence compression rate is reached.

The compression algorithm begins by removing determiners, time expressions and other low content words. More drastic compression rules are then applied to remove larger constituents of the parse tree until the required headline length is achieved. For the DUC 2004 headline generation task systems were required to produce headlines no longer than 75 bytes, i.e. about 10 words. The Figure 2.5 shows an example that helps to illustrate the sentence compression process.

Like the *trailing SBAR* rule, the other iterative rules identify and remove non-essential relative clauses and subordinate clauses from the lead sentence. A more detailed description of these rules can be found in [31] and [106]. In this example, we can see that after compression the lead sentence reads more like a headline.

### 2.2.3 Statistical Generation

The statistical approach toward title generation has been proposed and studied in the recent publications [101, 47, 99].

The basic idea of statistical approach is to first learn the correlation between the words in titles (title words) and the words in the corresponding documents (document words) from a given training corpus consisting of document-title pairs, and then apply the learned title-word-document-word correlations to generate titles for unseen documents [47].

Witbrock and Mittal [101] proposed a statistical framework for title generation where the task of title generation is decomposed into two phases, namely the title word selection phase and the title word ordering phase. In the phase of title word selection, each title word is scored based on its indication of the document content. During the title word ordering phase, the *appropriateness* of the word order in a title is scored using an n-gram statistical language model. The sequence of title words with the highest score in both title word selection phase and title word ordering phase is chosen as the title for the document. The follow-ups within this framework mainly focus on applying different approaches to the title word selection phase [47].

## 2.3 Related Machine Learning Methods

In this section, we will give a brief introduction to machine learning methods used in our research. We start by presenting some clustering methods which we have employed to develop a new text segmentation algorithm and segment combination algorithm. We then introduce Collin's incremental perceptron algorithm [26], which is used to learn HST generation models. Last, we discuss some semi-supervised learning methods, including a new method on using features derived from unlabeled data [66, 54, 53].

### 2.3.1 Incremental Perceptron

Collins et al. [25, 26] outlined a framework for linear models in natural language processing.

The task is to learn a mapping from inputs $x \in X$ to outputs $y \in Y$. For example, $X$ might be a set of documents, with $Y$ being a set of possible title. We assume:

- Training examples $(x_i, y_i)$ for $i = 1 \ldots n$.
- A function **GEN** which enumerates a set of candidates $\mathbf{GEN}(x)$ for an input $x$.
- A **representation** $\Phi$ mapping each $(x, y) \in X \times Y$ to a feature vector $\Phi(x, y) \in \mathbb{R}^d$.
- A **parameter vector** $\bar{\alpha} \in \mathbb{R}^d$.

The components **GEN**, $\Phi$ and $\bar{\alpha}$ define a mapping from an input $x$ to an output $F(x)$ through

$$F(x) = \arg \max_{y \in \mathbf{GEN}(x)} \Phi(x, y) \cdot \bar{\alpha} \tag{2.4}$$

where $\Phi(x, y) \cdot \bar{\alpha}$ is the inner product $\sum_s \alpha_s \Phi_s(x, y)$. The learning task is to set the parameter values $\bar{\alpha}$ using the training examples as evidence. The *decoding algorithm* is a method for searching for the arg max in Equation 2.4.

This framework is general enough to encompass several tasks in NLP. In this study, we are interested in title generation, where $(x_i, y_i)$, **GEN**, and $\Phi$ can be defined as follows:

- Each training example $(x_i, y_i)$ is a pair where $x_i$ is a document, and $y_i$ is its title.
- Given an input document $x$, $\mathbf{GEN}(x)$ is a set of possible titles for that document.
- The representation $\Phi(x, y)$ could track arbitrary features of document and title. For example, we could define the $i$-th component of the representation, $\Phi(x, y)$, to be whether or not the last word of the title appears in the document.

Algorithm 2.1 is the perceptron algorithm for parameter estimation. Note that the most complex step of the method is finding $z_i \leftarrow \arg \max_{z \in \mathbf{GEN}(x)} \Phi(x_i, z) \cdot \bar{\alpha}$, and this is precisely the decoding problem.

In [25, 26], they used the *averaged* parameters from the training algorithm in decoding test examples in their experiments. Say $\bar{\alpha}_i^t$ is the parameter vector after the $i$-th example is processed on the $t$-th pass through the data in the Algorithm 2.1. Then the averaged parameters $\bar{\alpha}_{\text{avg}}$ are defined as $\bar{\alpha}_{\text{avg}} = \frac{1}{NT}\sum_{i,t}\bar{\alpha}_i^t$.

**Algorithm 2.1:** A variant of the perceptron algorithm for structured prediction.

```
Input: $N$ training example $(x_i, y_i)$; the number of iterations $T$.
Output: Parameters $\bar{\alpha}$.
1 $\bar{\alpha} \leftarrow 0$
2 for $t \leftarrow 1 \dots T$ do
3     for $i \leftarrow 1 \dots n$ do
4         $z_i \leftarrow \arg\max_{z \in \mathbf{GEN}(x)} \Phi(x_i, z) \cdot \bar{\alpha}$
5         if $z_i \neq y_i$ then
6             $\bar{\alpha} \leftarrow \bar{\alpha} + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
7         end
8     end
9 end
```

Note that the difficulty of finding the arg max in Equation 2.4 is dependent on the interaction of **GEN** and $\Phi$. In many cases $\mathbf{GEN}(x)$ could grow exponentially with the size of $x$, making brute force enumeration of the members of $\mathbf{GEN}(x)$ intractable. For example, the number of possible titles for a document grows exponentially with the desired title length. Collins et al. [26] presents an alternative approach, the *incremental perceptron*, which is a variant on the structured perceptron, deals with the issue of the arg max may not be analytically vailable. It uses heuristic methods for finding arg max, replaces arg max by incremental *beam search* strategies (which returns a much smaller set of the candidates):

$$F(x) = \arg\max_{y \in \text{Top}(\mathbf{GEN}(x))} \Phi(x, y) \cdot \bar{\alpha} \tag{2.5}$$

Note that the incremental beam search is only a heuristic, there is no guarantee that this procedure will find the highest scoring parse. Search errors when

$$\arg\max_{y \in \mathbf{GEN}(x)} \Phi(x, y) \cdot \bar{\alpha} \neq \arg\max_{y \in \text{Top}(\mathbf{GEN}(x))} \Phi(x, y) \cdot \bar{\alpha} \tag{2.6}$$

In [26], they introduce two refinements including the *repeated use of a hypothesis* and the *early update*. The first refinement maintains a cache of examples and repeatedly iterates over them to update the model if the gold standard parse is not the best scoring parse from among the stored candidates (dynamically generate the constraints, i.e. incorrect parses, and uses these constraints to update the model while the original algorithm only looks at one constraint on each sentence and is extremely wasteful with the generated constraints implied by previously parsed sentences). Early-update aborts the search algorithm as soon as it has detected that an error has been made rather than allowing the parser to continue to the end of the sentence which leads to less noisy input to the parameter estimation algorithm; and also improve the efficiency.

### 2.3.2 Clustering

Clustering is an unsupervised learning problem which tries to group a set of points into clusters such as that points in the same cluster are more similar to each other than points in different clusters, under a particular clustering distortion or distance measure. There are two categorizations of clustering, e.g., hierarchical or partitional, depending on whether the algorithm clusters the data into a hierarchical structure or generates a flat partitioning of the data.

#### Hierarchical Clustering

In hierarchical clustering, the data is not partitioned into clusters in a single step. Instead, a series of partitions is created, which may run from a single cluster containing all objects to $n$ clusters each containing a single object. This gives rise to a hierarchy of clusters, also known as the cluster dendrogram. Hierarchical clustering methods can be further subdivided into two kinds of methods as follows.

**Divisive methods** create the cluster dendrogram in a top-down divisive fashion, starting with every data point in one cluster and splitting clusters successively according to some measure until a convergence criterion is reached, e.g., COBWEB [36], PDDP or principal direction divisive partitioning [16], and recursive cluster-splitting using a statistical transformation [32].

**Agglomerative methods** create the cluster dendrogram in a bottom-up agglomerative fashion, starting with each data point in its own cluster and merging clusters successively according to a similarity measure till a convergence criterion is reached. A typical example is hierarchical agglomerative clustering algorithm.

To illustrate hierarchical clustering, let us consider hierarchical agglomerative clustering in more detail.

#### Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering (HAC) is a bottom-up hierarchical clustering algorithm. In HAC, points are initially allocated to singleton clusters, and at each step the *closest* pair of clusters are merged, where closeness is defined according to a similarity measure between clusters. The algorithm generally terminates when a specified convergence criterion is reached. Different cluster-level similarity measures are used to determine the closeness between clusters to be merged—single-link, complete-link, or group-average [3].

Various HAC schemes have been recently shown to have well-defined underlying generative models: single-link HAC corresponds to the probabilistic model of a mixture of branching random walks, complete-link HAC corresponds to uniform equal-radius hyperspheres, whereas group-average HAC corresponds to equal-variance configurations [51]. The pseudo-code for HAC is given in Algorithm 2.2.

**Algorithm 2.2:** Hierarchical Agglomerative Clustering (HAC) algorithm

```
   Input: Set of data points $X = \{x_i\}_{i=1}^n, x_i \in \mathbb{R}^d$.
   Output: Dendogram representing hierarchical clustering of $X$.
1  Initialize clusters: Each data point $x_i$ is placed in its own cluster $C_i$. These clusters
   form the leaves of the dendogram, and constitute the set of current clusters.
2  repeat
3      Merge the two closest clusters $C_i$ and $C_j$ from current clusters to get cluster $C$.
4      Remove $C_i$ and $C_j$ from current clusters, add cluster $C$ to current clusters.
5      Add parent links from $C_i$ and $C_j$ to $C$ in the cluster dendogram.
6  until convergence
```

### Partitional Clustering

Let $X = x_{ii=1}^{n}, x_i \in \mathbb{R}^d$ be the set of $n$ data points we want to cluster. A partitional clustering algorithm generates a $K$-partitioning[2] of the data ($K$ given as input to the algorithm) by grouping the associated data points into $K$ clusters. Partitional algorithms can be classified into the following categories:

**Graph-theoretic based** These are discriminative clustering approaches, where an undirected graph $G = (V, E)$ is constructed from the data set each vertex $v_i \in V$ corresponding to a data point $x_i$ and the weight of each edge $e_{ij} \in E$ corresponding to the similarity between the data points $x_i$ and $x_j$ according to a domain-specific similarity measure. The $K$ clustering problem becomes equivalent to finding the $K$-mincut in this graph, which is known to be a NP-complete problem for $K > 3$. One class of methods for solving the graph partitioning problem take a real relaxation of the NP-complete discrete partitioning problem: these include spectral methods that perform clustering by using the second eigenvector of the graph Laplacian to define a cut [71]. The other class of methods use heuristics to find low-cost cuts in $G$: groups nodes based on the idea of defining neighborhoods using inter-connectivity of nodes in $G$, performs fast multi-level heuristics on $G$ at multiple resolutions to give good partitions, uses a modified cut criterion to ensure that the resulting clusters are well-balanced according to a specified balancing criterion [7].

**Density-based** These methods model clusters as dense regions and use different heuristics to find arbitrary-shaped high-density regions in the input data space and group points accordingly. Well-known methods include Denclue, which tries to analytically model the overall density around a point, and WaveCluster, which uses wavelet-transform to find high-density regions. Density-based methods typically have difficulty scaling up to very high dimensional data (more than 10,000 dimensions), which are common in domains like text [7].

[2] $K$ disjoint sets $\{X_k\}_{k=1}^K$ of $X$, whose union is $X$

**Mixture-model based** In mixture-model based clustering, the underlying assumption is that each of the $n$ data points $\{x_i\}_{i=1}^{n}$ to be clustered are generated by one of $K$ probability distributions $\{p_k\}_{k=1}^{K}$, where each distribution $p_k$ is the conditional distribution corresponding to the cluster $X_k$. The probability of observing any point $x_i$ is given by:

$$P(x_i|\Theta) = \sum_{k=1}^{K} \alpha_k p_k(x_i|\theta_k) \tag{2.7}$$

where $\Theta = (\alpha_1, \ldots, \alpha_k, \theta_1, \ldots, \theta_k$ is the parameter vector, $\alpha_1, \ldots, \alpha_k$ are the prior probabilities of the clusters, and $p_k$ is the probability distribution of cluster $X_k$ parameterized by $\theta_k$. The data generation process is assumed to be as follows: first, one of the $K$ components is chosen following their prior probability distribution $\{\alpha_k\}_{k=1}^{K}$; then, a data point is sampled following the distribution $p_k$ of the chosen component.

Since the cluster assignment of the points are not known, we assume the existence of a random variable $Y$ that encodes the cluster assignment $y_i$ for each data point $x_i$ and takes values in $\{1 \ldots K\}$. The goal of clustering in this model is to find the estimates of the parameter vector $\Theta$ and the cluster assignment variable $Y$ such that the complete log-likelihood of the data:

$$L(X, Y|\Theta) = \sum_{i=1}^{N} \log P(x_i, y_i|\Theta) \tag{2.8}$$

is maximized, where the i.i.d. (identically and independently distributed) assumption over the data points in $X$ leads to the factoring of the likelihood over the whole data set $X$ into individual probabilities over each data point $x_i$. Since $Y$ is unknown, the log-likelihood cannot be maximized directly. So, traditional approaches iteratively maximize the expected log-likelihood in the Expectation Maximization (EM) framework (Dempster et al., 1977). Starting from an initial estimate of $\Theta$, the EM algorithm iteratively improves the estimates of $\Theta$ and $p(Y|X, \Theta)$ such that the expected value of the complete-data log-likelihood is maximized, where the expectation is computed w.r.t. the posterior class distribution $p(Y|X, \Theta)$. It can be shown that the EM algorithm converges to a local maximum of the expected log-likelihood distribution (Dempster et al., 1977), and the final estimates of the conditional distribution $p(Y|X, \Theta)$ on convergence of the algorithm, are used to find the cluster assignments of the points in $X$.

Most of the work in this area has assumed that the individual mixture density components $p_k$ are Gaussian, and in this case the parameters of the individual Gaussians are estimated by the EM procedure. The popular K-means clustering algorithm [57] can be shown to be an EM algorithm on a mixture of $K$ Gaussians under certain assumptions. Another interesting model for Gaussian mixture model-based clustering is AutoClass [21], which also has a Bayesian model selection component for choosing the optimal number of clusters.

### 2.3.3 Semi-supervised Learning

In this section, we start by summarizing well-known semi-supervised learning methods in NLP. After that, we discuss about new semi-supervised learning method that uses features derived from unlabeled data.

#### Generative Maximum-Likelihood Models

Early research in semi-supervised learning for NLP made use of the EM algorithm for parsing [81] and part-of-speech tagging [65], but these results showed limited success. One problem with this approach and other generative models is that it is difficult to incorporate arbitrary, interdependent features that may be useful for solving the task at hand. Still, EM has been successful in some domains such as text classification [78].

#### Co-training and Bootstrapping

A number of semi-supervised approaches are based on the *co-training* framework [15], which assumes each instance in the input domain can be split into two independent views conditioned on the output class. A natural *bootstrapping* algorithm that follows this framework: train a classifier using one view of the labeled instances; use that classifier to label the unlabeled instances, which it is most confident about; train a classifier using the other view; use that classifier to label additional unlabeled instances, and so on, until all the unlabeled examples have been labeled. Similar varieties of bootstrapping algorithms have been applied to named-entity classification, parsing, and word-sense disambiguation. Instead of assuming two independent views of the data, Goldman and Zhou [38] uses two independent classifiers and one view of the data.

Both theoretical [15] and empirical analysis [77] have been done on co-training. Abney [1] analyzes variants of Yarowsky's bootstrapping algorithms [105] in terms of optimizing a well-defined objective.

#### Partitioning

Another class of methods, most natural in the binary case, view classification as partitioning instances—both labeled and unlabeled—into two sets. Suppose we define a similarity measure over pairs of instances and interpret the similarity as a penalty for classifying the two instances with different labels. Then we can find a minimum cut [14] or a normalized cut [89] that consistently classifies the labeled instances. Other methods based on similarity between instance pairs include label propagation, random walks, and spectral methods [2]. Transductive SVMs maximizes the margin of the examples with respect to the separating hyperplane [48].

#### Using features derived from unlabeled data

Each of the previous approaches attempts to label the unlabeled instances, either through EM, bootstrapping, or graph partitioning. A fundamentally different approach, which has been quite successful [66, 54, 53, 56] and is the approach we take, preprocesses the unlabeled data in a step separate from the training phase to derive features and then uses these features in a supervised model. This is normally called two-stage semi-supervised learning model. In our case, we derive word clustering and topic modeling from unlabeled data—a large collection of texts—and use the features in text segmentation, segment combination, and HST generation.

## 2.4 Summary

In this chapter, we presented the background knowledge for the research content in this thesis. We begin the chapter with the introduction to text segmentation approaches, which will be used in Chapter 4. We then review some typical title generation methods, which are related to our work in Chapter 5. In the last section, we summary some important machine learning methods used in our research, such as *incremental perceptron* for Chapter 5, *clustering* for Chapter 3 and Chapter 6, *semi-supervised learning* for Chapter 5.

# Chapter 3

# Supportive Knowledge

In this chapter, we present the supportive knowledge which is a kind of semantic knowledge that is employed in our research to provide semantic and topic information. We, firstly, give a short introduction to the supportive knowledge in the natural language processing field. We are then present two methods for acquiring supportive knowledge: word clustering and topic modeling. Next, we briefly describe the datasets used to acquire supportive knowledge. Finally, we present some experimental results with discussions.

## 3.1 Introduction

In the natural language processing (NLP) field, almost tasks have to use basic knowledge about document to get a "shallow" understanding about the document. It is usually encoded as features in the unsupervised or supervised learning model. The most popular feature types include lexical, position, part-of-speech surrounding context, and syntactic. They provide the information about words in a text at different levels. For example, lexical, position, and part-of-speech are information about a word, contextual features provide information about the relation between a word and its surrounding words, and syntactic features provide information about the grammatical interaction among words in a sentence. However, it is still limited in the small size of the context around a word, or in the boundary of a sentence. Consequently, those features only provide the local information of words, or local interaction between words in a document. In other words, NLP models which use those features cannot capture the relation between words in the wider range such as document, for example, collocation. In addition, another problem in NLP tasks is data sparsity[1], in which the model parameters for words that are rare in the labeled training data are poorly estimated. Furthermore, when the learning model is used for a new text, it cannot handle words that do not appear in the labeled training data. This phenomenon is called out-of-vocabulary (OOV) problem.

In this research, to reduce the effect of data sparsity and to capture the relation between words in the document range, we use supportive knowledge. It is a kind of semantic knowledge, which provide semantic and topic information about words and documents to support learning models. It is different from features based on words in that the sup-

[1]It has equivalent meaning to the word *sparseness*

portive knowledge based features have been acquired from a large collection of texts by unsupervised methods such as word clustering [18] and topic modeling [13]. Such methods can be used to represent word in the dense form because they group words into clusters or topics by category, semantic relation, or topic.

In the next sections, we investigate on two methods for acquiring supportive knowledge which are Brown word clustering [18] and Latent Dirichlet Allocation [13]. We then do experiments on a two large datasets WIKI and NIPS.

## 3.2 Word Clustering

In this section, we present a systematic introduction about the basic knowledge about word clustering. It is mostly employed from [20].

A typical word clustering algorithm partitions a set of words into classes or clusters [61]. Figure 3.1 shows a portion of a hierarchical clustering acquire from a small portion of text which contains 116 words in 11 sentences.

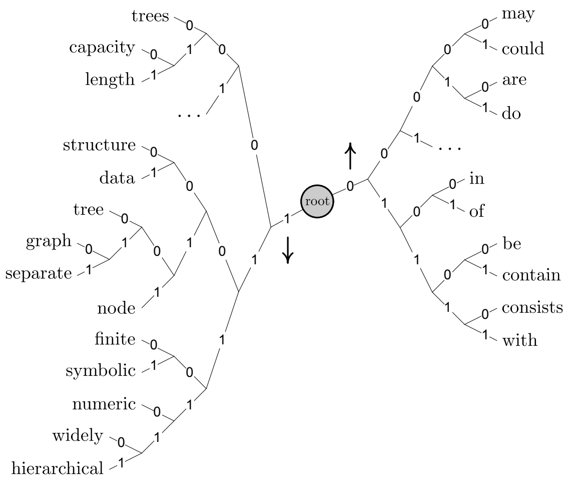


Figure 3.1: An example of a hierarchical clustering.

Grouping words (or anything else) into classes that reflect commonality of some property works as follows:

1. Defines the properties one cares about, and be able to give numerical values for each property;
2. Creates a vector of length $n$ with the $n$ numerical values for each item to be classified;
3. Viewing the $n$-dimensional vector as a point in an $n$-dimensional space, cluster points that are near one another.

This procedure leaves the following points open to variation:

1. The properties used in the vector;

2. The distance metric used to decide if two points are “close”;
3. The algorithm used to cluster.

Of the three aspects of our clustering procedure that we can vary, it is the first, the properties used, that seems to have the largest effect on the results, and we organize our discussion along this dimension. However, there is no doubt a lot of useful work to be done on choosing the proper metric and algorithm for clustering.

### 3.2.1 The Brown Clustering Algorithm

Brown et al. [18] proposed a bottom-up agglomerative word clustering algorithm to acquire a hierarchical clustering of words. The input of the algorithm is a large sequence of words $w_1, w_2, \ldots, w_n$ which are extracted from a large collection of texts. The output of the clustering algorithm is a binary tree, wherein a leaf represents a word. By choosing an internal node, we have a cluster containing the words in the corresponding sub-tree.

In [18], each word was characterized by the words that immedihtely followed it. More formally, $C(x)$ denotes the vector of properties of $x$ (intuitively, $x$'s “context”), in which $x$ is a word type. We can think of our vector for $w_i$ as counts, for each word $w_i$, of how often $w_i$ followed $w_i$ in the corpus:

$$C(w_i) = (|w_1|, |w_2|, \ldots, |w_n|) \tag{3.1}$$

Now there are many ways to define our distance measure on such vectors. Note that Euclidean distance would not be a good one, as words that occurred often would then be quite distant from rare words, even if they meant pretty much the same thing (and were used in the same way). For example, this would separate “fat” and “obese.” Normalizing by the count of $w_i$ would fix this, and then we would have a vector of conditional probabilities $P(w_j|w_i)$. However, what Brown et al. did was somewhat different.

To understand their metric we need to introduce a new concept, that of *mutual information.* We first define the mutual information $I(x;y)$ of two particular outcomes $x$ and $y$ as the amount of information one outcome gives us about the other. With the standard idea of information as $-\log P(x)$, this gives us:

$$I(x;y) = (-\log P(x)) - (-\log P(x|y)) = \log \frac{P(x,y)}{P(x)P(y)} \tag{3.2}$$

Suppose we want to know how much information the word “pancake” gives us about the following word “syrup.” The mutual information measure for this would be:

$$I(w_i = \text{pancake}; w_{i+1} = \text{syrup}) = \log \frac{P((w_i = \text{pancake}, w_{i+1} = \text{syrup}))}{P(w_i = \text{pancake})P(w_{i+1} = syrup)} \tag{3.3}$$

In our usual way, we abbreviate this as

$$I(\text{pancake}; \text{syrup}) = \log \frac{P(\text{pancake}, \text{syrup})}{P(\text{pancake})P(\text{syrup})} \tag{3.4}$$

A good way to get a feel for this measure is to see how it performs in various limits. For example, if "pancake" and "syrup" have no particular relation to each other then we would expect:

$$P(\text{syrup}|\text{pancake}) = P(\text{syrup}) \tag{3.5}$$

In this case

$$\begin{aligned} I(\text{pancake}; \text{syru}p) &= \log \frac{P(\text{pancake}, \text{syrup})}{P(\text{pancake})P(\text{syrup})} \\ &= \log \frac{P(\text{syrup}|\text{pancake})}{P(\text{syrup})} \\ &= \log \frac{P(\text{syrup})}{P(\text{syrup})} = 0 \end{aligned} \tag{3.6}$$

If they are perfectly coordinated then we get a very large number, as shown by:

$$\begin{aligned} I(\text{pancake}; \text{syrup}) &= \log \frac{P(\text{pancake}, \text{syrup})}{P(\text{pancake})P(\text{syrup})} \\ &= \log \frac{P(\text{pancake})}{P(\text{pancake})P(\text{syrup})} \\ &= \log \frac{1}{P(\text{syrup})} \end{aligned} \tag{3.7}$$

The *average mutual infonnation* of the random variables $X$ and $Y$, $I(X;Y)$, is defined as the amount of information we get about $X$ from knowing the value of $Y$, on the average. Thus its definition is the average over the mutual information of the individual combinations. (We assume that both random variables have possible values $\{w_1, w_2, \ldots, w_n\}$, although we could just as easily assume they have different possible values.)

$$I(X;Y) = \sum_{j=1}^{n} \sum_{i=1}^{n} P(w_i, w_j) I(w_i; w_j) \tag{3.8}$$

Average mutual information can also be formally defined using the notion of conditional entropy $H(X|Y)$ as follows:

$$H(X|Y) = -\sum_{j=1}^{n} \sum_{i=1}^{n} P(w_j)P(w_i|w_j) \log P(w_i|w_j) \tag{3.9}$$

$$\Rightarrow I(X;Y) = H(X) - H(X|Y) \tag{3.10}$$

As we cluster things (words) together, we lose specificity in our predictions and thus the average mutual information decreases. Obviously we would like this decrease to be as small as possible. Thus the metric used in [18] is the minimal loss of average mutual information. Suppose we are considering clustering the words "big" and "large" into a single group. We would first compute $I(w_i; w_{i-1})$ for the separate words. We would then create a class "big-large" whose vector $C(\text{big-large})$ is derived by summing the individual components of $C(\text{big})$ and $C(\text{large})$. We would then change all the other vectors, e.g., $C(\text{the})$, so that they have $n-1$ components rather than the original $n$ components (since they lost components for "big" and "large" but gained one for the group). The idea is to find groups in which the loss of mutual information is small. In general, the loss is smaller when the members of the group have similar vectors.

So we have specified what the study in [18] used for $C(w_i)$ and what it used as its distance metric. What remains is the algorithm by which the clusters were created, given this metric. If computational time were no object the algorithm would be trivial. Say we wanted 1,000 groups. We would just try all possible groupings into 1000 groups and pick the best on our metric. This is, of course, impossible in real life—there are too many groupings. A typical repair for this problem is to adopt a *greedy algorithm*. In this case, this means that the algorithm starts with $n$ clusters, one for each word. It then combines the two clusters that result in minimal loss of mutual information, and repeats until the desired final number of clusters is reached. This is not guaranteed to find the best clusters, but seems to work well in practice. However, in the study in question, with a vocabulary of 260,741 words, even this strategy was too expensive, and instead the algorithm defined 1,000 clusters initially, each containing one of the 1,000 most common words in the corpus, and then added each of the remaining words to one of these clusters using the greedy method. Note that in several cases the program correctly clustered misspellings with the properly spelled version of the word.

Brown et al. [18] tested these classes in two ways. They built a conventional trigram model and found a per-word cross entropy of 7.93 bits/word. The corresponding model using classes rather than words was measured at 8.08 bits/word: the accuracy has decreased, but the model has considerably fewer parameters. They also used the classes to smooth the trigram model and got a result of 7.88 bits/word. Thus, as one might expect, these classes do capture many of the same regularities as expressed by the trigram model.

### 3.2.2 Variants of Brown Clustering Algorithm

Martin et al. [63] extends the Brown clustering algorithm by using an objective function based on tri-gram models instead of bi-gram models. They also introduce an efficient exchange algorithm that tries to improve the objective by moving words from one cluster to another. The exchange technique was also used in [95].

### 3.2.3 Word Representation

To use the word clustering in our models, we encode each word cluster by a bit string that describes the path on the tree from the root to the cluster—the corresponding internal node. The path is encoded as follows: we start from the root of the tree with an empty bit string; “0” is appended to the bit string if we go *up*, and “1” is appended if we go *down*. For example, in Figure 3.1, {*trees, capacity, length*} is encoded by “100”, {*structure, data, tree, graph, separate, node*} is encoded by “110”, {*in, of*} is encoded by “010”, {*widely, hierarchical*} is encoded by “1111”, and so on. Each word in a cluster is represented by its cluster’s bit string.

If we view a cluster as an abstraction of a word, we can choose the arbitrary level of abstraction of a word in a hierarchical clustering. Indeed, we can use a prefix of a bit string as a representation of higher abstraction level. For example, in Figure 3.1, if we choose the prefix “11”, both words “tree” (110) and “hierarchical” (1111) become the member of a cluster at higher level abstraction.

This multi-level of abstraction in word representation helps our model can capture the semantic similarity of words at many levels. This is helpful in previous researches [53, 94].

## 3.3 Topic Modeling

Representing text corpora effectively to exploit their inherent essential relationship between members of the collections has become sophisticated over the years. Latent Semantic Analysis (LSA) [29] is a significant step in this regard. LSA uses a singular value decomposition of the term-by-document matrix to identify a linear subspace in the space of term weight features that captures most of the variance in the collection. This approach can achieve a considerable reduction in large collections and reveal some aspects of basic linguistic notions such as synonymy or polysemy. One drawback of LSA is that the resulting concepts might be difficult to interpret. For example, a linear combination of words such as “car” and “truck” could be interpreted as a concept “vehicle”. However, it is possible for the case in which the linear combination of “car” and “bottle” to occur. This leads to results which can be justified on the mathematical level, but which have no interpretable meaning in natural language.

Probabilistic Latent Semantic Indexing (pLSI) [45] was the successive attempt to capture the semantic relationship within a text. It relies on the idea that each word in a document is sampled from a mixture model, where mixture components are multinomial random variables that can be viewed as representation of *topics*. Consequently, each word is generated from a single topic, and different words in a document may be generated from different topics.

While Hofmann’s work is a useful step toward probabilistic text modeling, it suffers from severe over fitting problems [44]. Additionally, although pLSI is a generative model of the documents in the estimated collection, it is not a generative model of new documents. In other words, it is not clear how to assign probability to a document outside the training set [13]. The Latent Dirichlet Allocation (LDA) introduced by Blei et al. [13], is the solution to these problems. Since topic inference for new documents (based on an estimated topic model) is an important step in our proposal, LDA is a better choice than pLSI for this framework. Not only theoretical analysis, but also careful experiments have been conducted to prove the advantages of LDA over pLSA in [13].

There have been some other topic modeling methods proposed recently such as Dynamic Topic Model (DTM) [10], Correlated Topic Model (CTM) [11], and Topical N-gram

Model [100], which can be applied to the process of topic analysis. While still being able to capture rich relationships between topics in a collection, LDA is simpler than these models. For this reason, we choose LDA for the topic analysis step in our proposal. More details about LDA are given in the subsequent sections.

### 3.3.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [13, 39, 44] is a generative graphical model as shown in Figure 3.2. It can be used to model and discover underlying topic structures of any kind of discrete data in which text is a typical example. LDA was developed based on an assumption of the document generation process depicted in both Figure 3.2 and Algorithm 3.1.

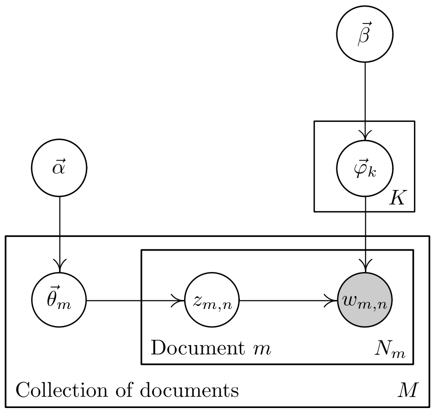


Figure 3.2: The generative graphical model of LDA.

We begin the presentation of LDA with some common notations:

- $M$: the total number of documents to generate (const scalar)
- $K$: the number of latent topics (const scalar)
- $V$: number of terms $t$ in vocabulary (const scalar)
- $\vec{\alpha}$: Dirichlet parameters
- $\vec{\theta}_m$: topic distribution for document $m$
- $\Theta = \left\{\vec{\theta}_m\right\}_{m=1}^{M}$: a $M \times K$ matrix
- $\vec{\phi}_k$: word distribution for topic $k$
- $\Phi = \left\{\vec{\phi}_k\right\}_{k=1}^{K}$: a $K \times V$ matrix
- $N_m$: the length of document $m$, here modeled with a Possion distribution with constant parameter $\xi$
- $z_{m,n}$: topic index of $n$-th word in document $m$

**Algorithm 3.1:** Generation process of LDA

```
1 foreach document $m \in [1, M]$ do
2     sample mixture proportion $\vec{\theta}_m \sim \mathrm{Dir}(\vec{\alpha})$
3     sample document length $N_m \sim \mathrm{Poisson}(\xi)$
4     foreach word $n \in [1, N_m]$ do
5         sample topic index $z_{m,n} \sim \mathrm{Mult}(\vec{\theta}_m)$
6         sample term for word $w_{m,n} \sim \mathrm{Mult}(\vec{\varphi}_{z_{m,n}})$
7     end
8 end
```

- $w_{m,n}$: a particular word for word placeholder $[m, n]$

The generative process can be interpreted as follows. A document containing $N_m$ words $\vec{w}_m = \{w_{m,n}\}_{n=1}^{N_m}$ is generated by first picking a distribution over topics $\vec{\theta}_m$ from a Dirichlet distribution $\mathrm{Dir}(\vec{\alpha})$, which determines topic assignments for words in that document. Then the topic assignment for each word placeholder $[m, n]$ is performed by sampling a particular topic $z_{m,n}$ from multinomial distribution $\mathrm{Mult}(\vec{\theta}_m)$. Finally, a particular word $w_{m,n}$ is generated for the word placeholder $[m, n]$ by sampling from multinomial distribution $\mathrm{Mult}(\vec{\varphi}_{z_{m,n}})$. The topics $\vec{\varphi}_k$ are sampled once for the entire corpus.

From the generative graphical model depicted in Figure 3.2, we can write the joint distribution of all known and hidden variables given the Dirichlet parameters as follows.

$$p\left(\vec{w}_m, \vec{z}_m, \vec{\theta}_m | \vec{\alpha}, \Phi\right) = \prod_{n=1}^{M} p\left(w_{m,n} | \vec{\phi}_{z_{m,n}}\right) p\left(z_{m,n} | \vec{\theta}_m\right) p\left(\vec{\theta}_m | \vec{\alpha}\right) \tag{3.11}$$

And the likelihood of a document $\vec{w}_m$ is obtained by integrating over $\vec{\theta}_m$ and summing over $\vec{z}_m$ as follows.

$$p\left(\vec{w}_m | \vec{\alpha}, \Phi\right) = \int p\left(\vec{\theta}_m | \vec{\alpha}\right) \prod_{n=1}^{M} p\left(w_{m,n} | \vec{\theta}_m, \Phi\right) d\vec{\theta}_m \tag{3.12}$$

Finally, the likelihood of the whole data collection $W = \{\vec{w}_m\}_{m=1}^{M}$ is the product of the likelihood of all documents:

$$p\left(W | \vec{\alpha}, \Phi\right) = \prod_{m=1}^{M} p\left(\vec{w}_m | \vec{\alpha}, \Phi\right) \tag{3.13}$$

## 3.3.2 Gibbs Sampling

### LDA Estimation

Parameter estimation for LDA by directly and exactly maximizing the likelihood of the whole data collection in Equation 3.13 is intractable. One solution is to use approximate estimation methods such as variational methods [13] or Gibbs sampling [39]. Gibbs sampling is a special case of Markov-Chain Monte Carlo (MCMC) and often yields relatively simple algorithms for approximate inference in high-dimensional models such as LDA.

Let $\vec{w}$ and $\vec{z}$ be the vectors of all words and their topic assignment of the whole data collection $W$. Gibbs sampling approach [39] is not explicitly representing $\Phi$ or $\Theta$ as parameters to be estimated, but instead considering the posterior distribution over the assignments of words to topics, $p(\vec{z}|\vec{w})$. We then obtain estimates of $\Phi$ and $\Theta$ by using this posterior distribution. In order to estimate the posterior distribution, Griffiths et al. [39] used the probability model for LDA with the addition of a Dirichlet prior on $\Phi$. The complete probability model is as follows.

$$\begin{aligned} w_i|z_i, \Phi^{(z_i)} &\sim \mathrm{Mult}(\Phi^{(z_i)}) \\ \Phi &\sim \mathrm{Dir}(\beta) \\ z_i|\Theta^{(d_i)} &\sim \mathrm{Mult}(\Theta^{(d_i)}) \\ \Theta^{(d_i)} &\sim \mathrm{Dir}(\alpha) \end{aligned}$$

Here, $\alpha$ and $\beta$ are hyper-parameters, specifying the nature of the priors on $\Theta$ and $\Phi$. These hyper-parameters could be vector-valued or scalar. The joint distribution of all variables given these parameters is $p(\vec{w}, \vec{z}, \Theta, \Phi|\alpha, \beta$. Because these priors are conjugate to the multinomial distributions $\Phi$ and $\Theta$, we are able to compute the joint distribution $p(\vec{w}, \vec{z})$ by integrating out $\Phi$ and $\Theta$.

Using this generative model, the topic assignment for a particular word can be calculated based on the current topic assignment of all the other word positions. More specifically, the topic assignment of a particular word $t$ is sampled from the following multinomial distribution.

$$p(z_i = k|\vec{z}_{\neg i}, \vec{w}) = \frac{n_{k,\neg i}^{(t)} + \beta_t}{\sum_{v=1}^{V}\left(n_k^{(v)} + \beta_v\right) - 1} \cdot \frac{n_{m,\neg i}^{(k)} + \alpha_k}{\sum_{j=1}^{K}\left(n_m^{(j)} + \alpha_j\right) - 1} \tag{3.14}$$

where

- $n_{k,\neg i}^{(t)}$ is the number of times the word $t$ is assigned to topic $k$ except the current assignment;
- $\sum_{v=1}^{V} n_k^{(v)} - 1$ is the total number of words assigned to topic $k$ except the current assignment;
- $n_{m,\neg i}^{(k)}$ is the number of words in document $m$ assigned to topic $k$ except the current assignment;

- $\sum_{j=1}^{K} n_m^{(j)} - 1$ is the total number of words in document $m$ except the current word $t$.

In normal cases, Dirichlet parameters $\vec{\alpha}$ and $\vec{\beta}$ are symmetric, that is, all $\alpha_k$ ($k = 1 \ldots K$) have the same value, and so do $\beta_v$ ($v = 1 \ldots V$).

After finishing Gibbs Sampling, two matrices $\Phi$ and $\Theta$ are computed as follows.

$$\phi_{k,t} = \frac{n_k^{(t)} + \beta_t}{\sum_{v=1}^{V} n_k^{(v)} + \beta_v} \tag{3.15}$$

$$\theta_{m,k} = \frac{n_m^{(k)} + \alpha_k}{\sum_{j=1}^{K} n_m^{(j)} + \alpha_j} \tag{3.16}$$

**LDA Inference**

Given an estimated LDA model, we can now perform topic inference for unknown documents by a similar sampling procedure as previous [44]. A new document $\hat{m}$ is a vector of words $\hat{\vec{w}}_m$; our goal is to estimate the posteria distribution of topics $\hat{\vec{z}}$ given the word vector $\hat{\vec{w}}$ and the LDA model $L(\Theta, \Phi)$: $p(\vec{z}|\vec{w}, L) = p(\hat{\vec{z}}, \hat{\vec{w}}, \vec{z}, \vec{w})$. Here, $\vec{w}$ and $\vec{z}$ are vectors of all words and topic assignment of the data collection upon which we estimate the LDA model. The similar reasoning is made to get the Gibbs sampling update as follows.

$$p(\hat{z}_i = k|\hat{\vec{z}}_{\neg i}, \hat{\vec{w}}; \vec{z}_{\neg i}, \vec{w}) = \frac{n_k^{(t)} + \hat{n}_{k,\neg i}^{(t)} + \beta_t}{\sum_{v=1}^{V}\left(n_k^{(v)} + \hat{n}_k^{(v)} + \beta_v\right) - 1} \cdot \frac{n_{\hat{m},\neg i}^{(k)} + \alpha_k}{\sum_{j=1}^{K}\left(n_{\hat{m}}^{(j)} + \alpha_j\right) - 1} \tag{3.17}$$

where the new variable $\hat{n}_k^{(t)}$ counts the observation of term $t$ and topic $k$ in new documents. This equation gives an illustrative example of how Gibbs sampling works: high estimated word-topic association $n_k^{(t)}$ will dominate the multinomial masses in comparison with the contributions of $\hat{n}_k^{(t)}$ and $n_{\hat{m}}^{(t)}$, the masses of topic-word associations are propagated into document-topic associations [44].

After performing topic sampling, the topic distribution of new document $\hat{m}$ is $\vec{\theta}_{\hat{m}} = \{\theta_{\hat{m},1}, \ldots, \theta_{\hat{m},k}, \ldots, \theta_{\hat{m},K}\}$ where each component is calculated as follows.

$$\theta_{\hat{m},k} = \frac{n_{\hat{m}}^{(k)} + \alpha_k}{\sum_{z=1}^{K} n_{\hat{m}}^{(z)} + \alpha_z} \tag{3.18}$$

## 3.4 Acquiring Supportive Knowledge

In the scope of this study, the supportive knowledge is acquired from some datasets. These datasets are, in turn, built from a large collection of plain texts which is easily collected from the Internet. In this section, we briefly describe the datasets. We also introduce some tools that are used to acquire supportive knowledge. Last, we discuss on the acquired supportive knowledge.

### 3.4.1 Datasets

In this research, we use two datasets for acquiring supportive knowledge: one is an outside dataset and another is an inside dataset.

- **WIKI** is a huge dataset that contains all Wikipedia articles in English, which had been created and updated until September 20, 2009. This dataset has a very large vocabulary, and its articles spread over many domain (817,858 categories). It is good for acquiring word clustering, which need as many word contexts as possible, and topic modeling, which need as many document as possible.
- **ALG** is a small dataset that contains all sections of the textbook "*Introduction to Algorithms*" [27]. This dataset is used for acquiring topic modeling, which is, in turn, used as an in-domain topic model (computer science) for experiments in Chapter 5.

The details information of above datasets are described in Appendix B.

### 3.4.2 Tools

To acquire word clustering and topic modeling, we use the following tools.

- **WCluster**: This tool implements the Brown clustering algorithm [18]. It has been written in C by Percy Liang [54]. The source code is open and freely available on the author's page[2].
- **GibbsLDA++**: This tool implements a collapsed Gibbs sampling version of LDA. It has been written in C++ by Phan et al. [83]. The source code is open and freely available on the SourceForge[3].

### 3.4.3 Data Transformation and Preprocessing

#### Data Transformation

The Wikipedia articles are formatted by the Wikitext Language[4]. They are transformed to plain texts by our own toolkit named **Wikipedia Processing Toolkit**. This tool reads a dump file of the Wikipedia and transforms Wikitext articles inside that file to plain text articles.

[2]`http://www.eecs.berkeley.edu/~pliang/software/`
[3]`http://gibbslda.sourceforge.net/`
[4]`http://en.wikipedia.org/wiki/Wikitext`

**Data Preprocessing**

The pre-processing process are done on all datasets with the following steps.

1. *Sentence boundary detection* is the process of splitting every document into sentences. It is required to make the input for the word clustering tool.
2. *Tokenization* is the process of detaching marks from words in a sentence. It is required for every NLP task.

In this research, we do not need the case-sensitive feature of a word. Therefore, we convert all the words to be lowercase in the datasets.

For word clustering, the surroundings of a word is its context. Therefore, we keep every word to make the input. In the other hand, topic modeling is very sensitive to the popular words and rare words [44]. Moreover, the complexity of the topic modeling algorithm depends on the size of the vocabulary. Therefore, we should reduce the size of the vocabulary and remove the popular words and rare words. In our experiments, we use the following heuristic rules[5]:

1. Remove terms occurring in fewer than 0.1% of all documents;
2. Remove terms occurring in more than 50% of all documents.

The criterion (1) will remove a large number of rare words in the vocabulary. Meanwhile, the criterion (2) will remove only some popular words, but it saves time during inference when we use Gibbs sampling. The reason is that a topic assignment must be sampled for each instance of each term and the terms removed in (2) tend to be very frequent.

## 3.5 Experiments

### 3.5.1 Word Clustering

We have run the Brown clustering algorithm on the WIKI dataset to acquire two word clusters with the number of clusters is 512 and 1,024, respectively. The reason is that, the more clusters, the better and more stable learning model [94]. A word clustering that contains 1,000 clusters is the most popular design [66, 54, 53, 94]. We acquire a clustering of 512 clusters to test the performance of the model in different clustering designs.

In Table 3.1 (page 41), we show some word clusters in 1024 clusters that are acquired from the WIKI dataset. Each row is a cluster, in which the left cell contains a bit string representing the cluster and the right cell contains top 25 words of the cluster. The prefix of the bit string is underlined to show that words in the clusters with the same prefix have semantically similarity at the corresponding level. For example, the clusters at line

[5]Thanks to Prof. David M. Blei for sharing these rules on the mailing list of topic modeling.

3 and line 4 in the Table 3.1 have the same prefix at length 9. So, in addition to the full bit string representation, such as "10001010011" for "computer" or "10001010010" for multimedia, we can use the higher abstraction level by using "100010100" to represent for both words. Thereby, although two above words belongs to two different clusters, we can still capture the similarity between those words.

Furthermore, because the word clustering has been acquired from a large collection of text, it could address the problem of OOV in the learning model. For example, we assume that the word "dangerously" does not occur in the training data, but the word "amazingly" occur in the training data. By using clustering-based representation, the learning model has an estimated parameter for "100000100", therefore, it can handle the word "dangerously" in the testing process.

Table 3.1 also shows that some clusters containing words that functionally related (adverbs in line 1 and 2). Meanwhile, some other clusters containing words that topically related (movie in line 8), and so on.

Other oservation on word clustering [56] shows that the window size in representing contextual information of a word has an interesting effect on the types of clusters. With larger windows, the clusters tend to be more topical, whereas smaller windows result in categorical clusters. In our experiments, we use the window size of 1 as normally used in other studies.

### 3.5.2 Topic Modeling

Choosing the number of topics $K$—a long-discussed subject—has a huge effect on the learned topics. While the arrival of the non-parametric topic model [9] provided a neat mathematical solution to this problem, it was less widely adopted in practice. Wallach et al. [98] examined that if LDA has sufficient topics to model $W$ well, the assignments of tokens to topics should be relatively invariant to an increase in $K$—i.e., the additional topics should be seldom used. For example, if 10 topics is sufficient to accurately model the data, then increasing the number of topics to 20 shouldn't significantly affect inferred topic assignments. In other words, if the performance of the model is maximized at a specific value of $K$, the larger value of $K$ has no effects. However, the complexity of the topic modeling strongly depends on the number of topics [92]. Therefore, we need to choose an appropriate value of $K$ for each dataset. Moreover, the number of topics $K$ also affects on the our learning model, such as in title generation task.

In our experiments, WIKI dataset is a huge dataset with a broad range of topics. Therefore, the number of topics $K$ should be very large (hundreds to thousands). To investigate the effect of the number of topics, we choose $K = 200$ (average) and $K = 1000$ (large). In Table 3.2 (page 3.2) and Table 43 (page 43), we show some sample topics with top-ten most likely words. The topic number is on the left column and starts from 0. Each row in the right column is the top-ten most likely words of the corresponding topic.

To reduce the number of topics $K$ and acquire fine-grained topic models, we have estimated LDA on ALG dataset. Another reason is that, we want to investigate the effect of the in-domain topic model. In experiments, we choose $K = 100$.

## 3.6 Summary

In this chapter, we firstly presented two methods for acquiring supportive knowledge such as word clustering and topic modeling. We then give a briefly introduced to the Brown clustering algorithm [18], and latent Dirichlet allocation [13]. Next, above algorithms have been run on the two datasets WIKI and ALG to acquire word clustering and topic modeling with the different number of clusters. Finally, we presented the results of our experiments with some discussion. In next chapters, we will integrate this acquired supportive knowledge in the learning models.

Table 3.1: Sample word clusters acquired from WIKI dataset.

| **Bit string** | **Sample words** |
|---|---|
| 100000100 | increasingly equally sufficiently unusually surprisingly overly environmentally inherently immensely statistically infinitely dangerously enormously arbitrarily excessively finitely architecturally terribly aesthetically amazingly abnormally geologically disproportionately ecologically intrinsically ... |
| 100000101 | relatively extremely fairly reasonably exceptionally comparatively moderately remarkably incredibly hugely notoriously exceedingly mildly extraordinarily strikingly terminally mobb wonderfully ogc chronically downright self-inflicted prohibitively deceptively more-or-less ... |
| 10001010011 | drug computer chemical pc mechanical quantum java molecular linux differential real-time unix computational hydraulic planetary forensic multiplayer graphical c++ ceramic semiconductor polymer midi capitalist sql ... |
| 10001010010 | digital commercial global mobile domestic satellite cable corporate virtual retail consumer wireless google recreational portable cooperative promotional luxury specialty stereo terrestrial premium seasonal cellular multimedia ... |
| 1011011110000 | translation language dialect spelling alphabet wikipedia pronunciation accent cyrillic phonology orthography transliteration immersion subtitles romanization pidgin cricketarchive fluently angelou sheepdog snares folktale syllabary demonym patois ... |
| 1011011110001 | revolution empire dynasty descent mythology idol ancestry monarchy cuisine nadu isles rite catholicism diaspora ssr armada rhapsody guiana inquisition numerals polynesia franc riviera shogunate numeral ... |
| 10110111101111 | government constitution railways treasury judiciary populace taxpayer govt mujahideen chancellery doj sarbanes-oxley magistracy sebi government. papists nomenklatura rowlatt gramm-leach-bliley disd glass-steagall heimwehr mujahedeen trivialization epbc ... |
| 101111100100 | movie film porn imdb telenovela pinot glyndebourne telefilm phonographic dreamcoat ravinia pooram womad bamboozle mid-autumn fishtank sziget pinkpop film. tv-film feature-film mechanix qingming bumbershoot tanabata ... |
| 101111100000 | pop jazz dance blues folk r&b hip hop rap hip-hop indie disco hardcore reggae surf tango trance playback cabaret graffiti avant-garde bluegrass oldies techno salsa ... |

Table 3.2: The top-ten most likely words of a topic modeling on WIKI with $K = 200$.

| **Topic** | **The most likely words** |
|---|---|
| 0 | minister president office secretary prime chief appointed post affairs served |
| 1 | division army regiment infantry corps unit battalion brigade units artillery |
| 3 | students education science program student programs courses studies academic faculty |
| 14 | radio station channel television news fm broadcasting broadcast stations network |
| 20 | russian soviet russia moscow union alexander ukraine ukrainian scout vladimir |
| 25 | british london uk britain royal bbc turkish kingdom turkey england |
| 32 | medical hospital health disease medicine cancer blood care patients treatment |
| 40 | school high schools students elementary class district middle public grade |
| 46 | united states u.s. american america kingdom canada u.s americans national |
| 52 | party election liberal parliament conservative elected member labour elections political |
| 64 | isbn published poetry writer writing literature works books literary book |
| 71 | series comics comic doctor character dc marvel story batman strip |
| 80 | son prince married daughter died duke queen wife death father |
| 89 | football game nfl bowl season yards field pass defensive back |
| 97 | training master skills practice bond work techniques experience technique martial |
| 104 | station line railway railroad train trains rail service lines stations |
| 111 | village district county poland voivodeship gmina administrative lies approximately regional |
| 125 | tree plant garden plants trees native leaves flowers rose flower |
| 139 | club castle football clubs sports founded history sport ground based |
| 150 | bank business company financial stock management companies million firm insurance |
| 162 | music records record label rock video sound artists dj pop |
| 176 | album single released chart song singles track version billboard uk |
| 185 | world open championships championship tournament tour won champion junior cup |
| 192 | time years made began left back year continued returned early |
| 195 | engine speed design built engines fuel designed type production weight |

Table 3.3: The top-ten most likely words of a topic modeling on WIKI with $K = 1000$.

| **Topic** | **The most likely words** |
|---|---|
| 9 | began years time early continued work year success helped interest |
| 63 | dark aka night man midnight shadows black darkness death secret |
| 86 | sun stars star galaxy cluster constellation magnitude spiral galaxies ngc |
| 104 | university professor harvard stanford berkeley ph.d. american california yale school |
| 161 | circuit output voltage current input power circuits tube loop amplifier |
| 195 | conference tournament ncaa team basketball championship men teams university won |
| 218 | water supply surface pool drinking fresh swimming deep waters sanitation |
| 222 | birds bird eagle hawk owl common johnston falcon pigeon dove |
| 225 | year years time success left returned end back successful return |
| 237 | children child parents birth age baby parent adult adults infant |
| 244 | tells back asks finds takes find room sees begins leaves |
| 266 | news reporter anchor weather morning sports abc television network weekend |
| 284 | press university pp. ed. oxford vol cambridge history studies london |
| 328 | army general military officer commander colonel lieutenant rank officers command |
| 398 | politician player american writer author actor poet british footballer canadian |
| 444 | film directed starring drama cast comedy ralph based written nancy |
| 531 | electric current magnetic electrical battery ac motor charge power wire |
| 563 | contract pay paid employees fee employee contracts money payment fees |
| 651 | computer computers software ibm technology systems electronic computing system program |
| 664 | problem algorithm graph problems node solution set algorithms nodes number |
| 680 | earth solar observatory mars moon telescope sun planet astronomy astronomical |
| 739 | emergency rescue service response ambulance aid medical equipment volunteer safety |
| 788 | magazine editor journalist times published journalism column writer issue news |
| 979 | function matrix functions vector operator defined integral real linear form |
| 981 | christmas donald eve holiday carol duck special december gift santa |

# Chapter 4

# Text Segmentation

This chapter presents our work on the text segmentation task. In the first section, we introduce our approach on text segmentation. We then present the non-systematic semantic relation, which is a type of relationship in lexical cohesion. In this research, we use supportive knowledge to recognize that relation to improve the performance of the text segmentation task. Next, the details of our model using supportive knowledge has been described. To examine our proposed model, we do experiments on the widely-used public dataset. We finish this chapter with some discussion on the experimental results.

## 4.1 Introduction

Text segmentation is one of the fundamental problems in natural language processing with applications in information retrieval, text summarization, information extraction, and so on [50]. It is a process of splitting a document or a continuous stream of text into topically coherent segments. Text segmentation methods can be divided into two categories by the structure of output that is linear segmentation [22, 34, 42, 46, 59, 87, 96] and hierarchical segmentation [79], or by the algorithms that are unsupervised segmentation or supervised segmentation. In this research, we focus on the unsupervised-linear text segmentation method. The main advantage of unsupervised approach is that it does not require labeled data and is domain independent.

Almost unsupervised text segmentation methods are based on the assumption of cohesion [41], which is a device for making connection between parts of the text. Cohesion is achieved through the use of reference, substitution, ellipsis, conjunction, and lexical cohesion. The most frequent type is lexical cohesion, which is created by using semantically related words. Halliday and Hasan in [41] classified lexical cohesion into two categories: reiteration and collocation. Reiteration includes word repetition, synonym, and superordinate. Collocation includes relations between words that tend to co-occur in the same contexts, which are the systematic and the non-systematic semantic relations.

The current approaches in lexical cohesion-based text segmentation only focus on the first category of lexical cohesion, reiteration. Most of them use reiteration with the assumption that the repetition of words can play as the indicator of the topic coherence in a segment and the topic incoherent between segments. By using reiteration, those approaches can compute the semantic relation between two blocks of texts via some similarity-distance measurement to determine whether they can put a segment boundary between those blocks.

The collocation is the most problematical part in lexical cohesion [41, 68, 87]. It includes the semantic relation between words that tend to co-occur. Morris and Hirst in [68] first tried to take into account the collocation in text structuring. However, they can only make some manual experiments on the text due to the limitation of available electronic resources at that time. In [6], they try to use WordNet as a device for recognizing synonym and hyponym in text segmentation as an intermediate step to summarize a text. The resource-based approach has some limitations. For instance, WordNet mainly contains relations between nouns and is not available for almost languages. On the other hand, WordNet or thesauri normally contain relation between words which can be recognized without context such as {*apple, orange, fruit*}. In other words, those approaches can only take into account the systematic semantic relation.

In this research, we investigate the way to recognize the second relation in collocation, *non-systematic semantic relation*, in order to improve the text segmentation performance. This relation holds between two words or phrases in a discourse when they pertain to a particular theme or topic, which is normally hard to classify without context. For instance, {*paper, contribution, review*} in *conference* topic or {*translation, word, meaning*} in *language* topic are examples of classes of non-systematic semantic relation. Due to the nature of that relation, a topic model [13, 29, 45] estimated based on the co-occurence of words would be appropriate for recognizing it. In the scope of this paper, we attempt to use Latent Dirichlet Allocation (LDA) [13], which has many advantages and is widely adopted in comparison to previous topic model methods such as Latent Semantic Analysis (LSA) [29] or Probabilistic Latent Semantic Indexing (pLSI) [45]. The LDA model used in this research is estimated from a very large copora which contains all the articles of Wikipedia—the free encyclopedia.

## 4.2 Non-systematic Semantic Relation

Morris and Hirst [68] were the first to apply lexical cohesion for text segmentation. Based on the reiteration and collocation relationships in [41], they divided lexical cohesion into five types of relationships that are presented in Appendix A. The reiteration includes not only identity of reference or word repetition, but also the use of synonym or superordinate. The collocation includes semantic relationships between words that often co-occur. They can be further divided into two categories of relationship: systematic semantic and non-systematic semantic.

A systematic semantic relation holds between words or group of words that can be classified in a fairly straightforward way. For instance, it includes antonyms, members of an ordered set such as {*one, two, three*}, members of an unordered set such as {*red, green, blue*}, or part-to-whole relationships like {*eyes, mouth, face*} [68].

A non-systematic semantic relation holds between words that tend to occur in similar lexical environments, in which, they describe things that tend to occur in similar situations or contexts. In other words, they normally belong to a particular theme or topic. In the

The Hubble$^{680}$ Space$^{680}$ Telescope$^{680}$, one of the most important$^{375}$ telescopes$^{680}$ ever built$^{272}$, will help astronomers$^{680}$ search$^{253}$ for advanced$^{365}$ life$^{229}$ in space$^{680}$ and may find$^{664}$ an answer$^{973}$ to the age-old$^{617}$ question$^{973}$: are we alone in the universe$^{253}$.

The information$^{973}$ collected$^{827}$ by the Hubble$^{680}$ will be able to test$^{905}$ the common$^{299}$ assumptions$^{617}$ that we live$^{365}$ on an average$^{851}$ planet$^{680}$ orbiting$^{86}$ an average$^{778}$ star$^{86}$, that our solar$^{680}$ system$^{820}$ must be typical$^{851}$ of others throughout the galaxy$^{86}$, and that many advanced$^{868}$ life$^{229}$ forms$^{224}$ have evolved$^{375}$ in the universe$^{253}$.

Analysis$^{827}$ of data$^{827}$ sent back$^{713}$ over the last 30 years$^{713}$ by unmanned$^{680}$ spacecraft$^{680}$ from distant$^{680}$ regions$^{86}$ of the solar$^{86}$ system$^{820}$ is already seriously questioning$^{973}$ these assumptions$^{617}$.

The way the earth$^{224}$ evolved$^{874}$ holds$^{868}$ the key$^{365}$ to the question$^{973}$ of life$^{229}$ in space$^{680}$.

Images$^{680}$ of Mars$^{253}$, and radar$^{680}$ pictures$^{973}$ of Jupiter$^{680}$, Saturn$^{680}$, Uranus$^{680}$ and Neptune$^{680}$ show$^{571}$ that our earth$^{253}$ and its moon$^{680}$ are unique$^{664}$—at least in the solar$^{680}$ system$^{820}$ .

Figure 4.1: The first segment of the article "Stargazers" has been topic-assigned

other hand, words in this relation can have different part-of-speeches. For instance, some classes are {*paper, conference, review, presentation*} or {*language, translate, speak*}. This type of relationship is the most problematic, especially from a knowledge representation point of view. It normally holds between words in a specific context [68].

In this research, to take into account the non-systematic semantic relation, we employ a topic model. A topic model is usually estimated based on the co-occurrence of words in a large collection of documents. In the scope of this research, we use latent Dirichlet allocation (LDA) [13]. A brief description of LDA has been presented in Section 3.3. In LDA, every word is grouped into topics with specific probabilistic, and a word could belong to several topics with different probabilistic. The inference process of LDA will assign a topic to each word in a document.

In the real world, there are many ways to combine a group of words in a non-systematic semantic relation. If a topic model would like to assign appropriate topics to words in a text, it should be estimated from a collection that contains documents in which those words co-occur. Therefore, to cover almost co-occurrence of words, we need to estimate the topic model on a very large collection of texts, and that collection should be also topical-balanced. In our research, we estimate a LDA model from the collection of articles in Wikipedia, which should satisfy our requirements.

In Figure 4.1, we show an inferred portion of a text with topics for content words. It is a well-known example in text segmentation entitled "Stargazers" [42]. In that example, the topic number of every word is superscripted.

In the above example, we can see some group of words that are topical-related. They are also assigned the same topic number. For instance, some typical groups are {*Hubble, telescopes, astronomers, planet, spacecraft*} in topic 680, {*orbiting, star, galaxy*} in topic

86, {*search, Mars, universe*}, and so on.

To make the example more illustrative, in Table 4.1, we show the top 20 most likely words of some topics corresponding to the assigned topics in the example in Figure 4.1.

Table 4.1: Top 20 most likely words of the topic model estimated on Wikipedia. The topics listed according to the topics in Figure 4.1.

| **Topic** | **Top 20 most likely words** |
|---|---|
| 86 | sun stars star galaxy cluster constellation magnitude ngc galaxies spiral dwarf light orion hd years sky apparent zodiac approximately nebula ... |
| 224 | rocks rock volcanic formation formed volcano geology geological lava eruption basin deposits fault plate ago earth surface cone found volcanoes ... |
| 253 | earth planet space alien ship universe aliens worlds galaxy race planets science travel mars technology series destroyed ships spaceship fiction ... |
| 365 | human humans world race humanity beings life civilization future created nature technology people artificial form living body advanced natural survival ... |
| 680 | earth observatory solar mars moon telescope planet sun astronomical astronomy jupiter venus orbit planets comet astronomer observations saturn system planetary ... |
| 827 | phase analysis pattern patterns information methods data phases based structure determine identify study identification method specific techniques important activity detection ... |

## 4.3 Text Segmentation with Supportive Knowledge

In this research, we follow the general framework presented in Section 2.1. We integrate the topical information into the contextual representation and the similarity computation step. In smoothing step, we use the anisotropic diffusion technique on the image representation of the similarity matrix to reduce noise in homogeneous regions, make homogeneous regions more homogeneous, and sharpen the boundaries between homogeneous regions [46]. In segmentation step, we re-implement the minimum cut segmentation algorithm, which has been normally applied to image segmentation problem [89]. We follow the dynamic programming algorithm in [59] to find an exact solution for the minimum cut problem in text segmentation.

### 4.3.1 Similarity Computation

In our model, the similarity between two sentences $s_i$ and $s_j$ is a linear combination of two similarity measure: lexical-based and topical-based.

The lexical-based similarity is the cosine of two vectors that represents word frequency in two sentences. To reduce the effect of common words in general English text, we use a vocabulary without stopwords, which does not play any role in semantic relation. In the other hand, this representation model cannot address the issue of words that are not in stopwords list but occur throughout a text in a particular subject. Those words may play an important role in understanding the meaning of a text, but it makes no effect on the text segmentation task. For instance, the word "earth" occurs in almost sentences in the text "Stargazers" in Fig. 4.1. Thus, its occurrence does not mark a change in topic. To address this issue, we employ a modified version of TF-IDF[1] weighting score [22, 59], in which we split a text to chunks which are treated as "documents". At the end, the lexical-based similarity between two sentences is computed as follows:

$$\begin{aligned}\mathrm{sim}_{\mathrm{lex}}(s_i, s_j) &= \cos(\text{tf–idf}_{s_i}, \text{tf–idf}_{s_j}) \\ &= \frac{\sum_{v \in V} \text{tf–idf}_{v,s_i} \times \text{tf–idf}_{v,s_j}}{\sqrt{\sum_{v \in V} \text{tf–idf}^2_{v,s_i}} \times \sqrt{\sum_{v \in V} \text{tf–idf}^2_{v,s_j}}}\end{aligned} \tag{4.1}$$

where $v$ is a word in the vocabulary $V$. In practice, we use an exponential version of the above similarity to accentuate differences between low and high lexical similarities $e^{\mathrm{sim}(s_i,s_j)}$.

The topical-based similarity between two sentences is computed based on topic distribution of those sentences which is, in turn, computed by (3.18). Previous studies [13, 39] normally use Kullback–Leibler divergence (KL) for computing the similarity. However, the KL divergence is not a distance measure proper because it is not symetric. Thus, alternatively, we use information radius (IRad) [61] as the topical-based similarity measure, which is also known as Jenshen–Shanon divergence (JS)—a variation of Kullback–Leibler divergence.

$$\mathrm{IRad}(p, q) = \mathrm{JS}(p\|q) = \mathrm{KL}\left(p \,\middle\|\, \frac{p+q}{2}\right) + \mathrm{KL}\left(q \,\middle\|\, \frac{q+p}{2}\right) \tag{4.2}$$

JS is symmetric ($\mathrm{JS}(p\|q) = \mathrm{JS}(q\|p)$) and there is no problem with infinite values since $\frac{p_i+q_i}{2} \neq 0$ if either $p \neq 0$ or $q \neq 0$. $\mathrm{JS}(p\|q)$ ranges from 0 for identical distributions to $2\log 2$ for maximally different distributions. The JS divergence is transformed to the similarity measure as follows [61]:

$$\mathrm{sim}_{\mathrm{topic}}(s_i, s_j) = \mathrm{IRad}(s_i, s_j) = 10^{-\beta \mathrm{JS}(p_{s_i}\|p_{s_j})} \tag{4.3}$$

where $\beta$ is a parameter that can be tuned for optimal performance. In practice, we normally choose $\beta = 1$.

The final similarity score between two sentences is the linear combination of (4.1) and (4.3), which is computed as follows:

$$\mathrm{sim}(s_i, s_j) = \lambda \mathrm{sim}_{\mathrm{lex}}(s_i, s_j) + (1-\lambda)\mathrm{sim}_{\mathrm{topic}}(s_i, s_j) \tag{4.4}$$

[1]TF-IDF: Term Frequency–Inverse Document Frequency

where $\lambda$ is a model's parameter, which can be tuned based on development set.

To reduce noise and reduce the number of edges in the graph that represents the similarity matrix, we use a threshold for the similarity score. This threshold is also optimized based on the development set.

To represent the relation between all pairs of sentences in a text, we use the graph-based approach. We build a graph $G = (V, E)$, in which a vertex $v_i$ represents the $i$-th sentence and an edge $e_{i,j}$, which connects vertex $v_i$ and vertex $v_j$, represents the similarity between the $i$-th sentence and $j$-th sentence $\text{sim}(i, j)$. In implementation, the graph is represented as a similarity matrix. We can then use DotPlot to visualize the matrix.

### 4.3.2 Smoothing Technique

For short text segments, the similarity score between two sentences $\text{sim} s_i, s_j$ is unreliable. One sentence that contains only common words and rare words may make a sudden shift in similarity scores. Without proper smoothing, these cases will lead the system astray. In this research, we employ the exponentially weighted moving average (EWMA) smoothing technique, which is developed by [88, 58]. The word counts vector of a sentence is changed by adding counts of words that occur in adjoining sentences. These counts are weighted in accordance to their distance from the current sentence. For example, the changed vector $\hat{s}_i$ of the word counts vector $s_i$ is computed as follows:

$$\hat{s}_i = \sum_{j=0}^{k-1} e^{-\delta j} s_{i+j} \tag{4.5}$$

where $k$ is the window size and $\alpha$ is a parameter that controls the degree of smoothing.

### 4.3.3 Decoding

In decoding step, almost systems employ clustering techniques [87, 22, 23, 59] for splitting a text into topically coherent segments. There are two main clustering methods are used.

#### Divisive Hierarchical Clustering

A text segment is defined by two sentences $i$ and $j$ (inclusive). This is represented as a square region along the diagonal of the similarity matrix. Let $s_{i,j}$ denote the sum of the similarity values in a segment and $a_{i,j} = (j - i + 1)$ be the inside area of the region. $B = b_1, b_2, \ldots, b_m$ is a list of $m$ coherent text segments. $s_k$ and $a_k$ refers to the sum of rank and area of segment $b_k \in B$. The inside density of $B$ denoted by $D$ is computed as follows:

$$D = \frac{\sum_{k=1}^{m} s_k}{\sum_{k=1}^{m} a_k} \tag{4.6}$$

To initialize the process, the entire document is placed in $B$ as one coherent text segment. The process splits one of the segments in $B$ at each step. The criterion of selecting a point for splitting is maximizing the density $D$.

One of the advantages of this approach is the ability of choosing the number of segments. In [87, 22, 23], they define the density change, or gradient $\Delta D^{(n)} = D^{(n)} - D^{(n-1)}$, where $D^{(n)}$ is the density of $n$ segments. At some step, if $\Delta D^{(n)}$ is below the threshold, the split process is stopped.

### Spectral Clustering

This approach is based on the idea of a normalized minimum cut on a graph [89]. First, let us define some notation.

- $G = (V, E)$ is the graph with vertices $v_i$ and edges $e_{i,j}$ $(i, j = 1 \ldots N)$, where $N$ is the number of vertices. Each edge $e_{i,j}$ has weight $w_{i,j}$. The graph is created based on the similarity matrix of the text.
- $\mathrm{vol}(A) = \sum\limits_{u \in A, v \in V} w_{u,v}$ is the volume of a subset $A \subset G$.
- $\mathrm{assoc}(A) = \sum\limits_{u \in A, v \in A} w_{u,v}$ is the association of a subset $A \subset G$.
- $\mathrm{cut}(A, B) = \sum\limits_{u \in A, v \in B} w_{u,v}$ is the value of the cut that split $G$ in to two separate subsets $A$ and $B$.
- $\mathrm{Ncut}(A, B) = \dfrac{\mathrm{cut}(A, B)}{\mathrm{vol}(A)} + \dfrac{\mathrm{cut}(B, A)}{\mathrm{vol}(B)}$ is the normalized cut value defined in [89].
- $\mathrm{Nassoc}(A, B) = \dfrac{\mathrm{assoc}(A)}{\mathrm{vol}(A)} + \dfrac{\mathrm{assoc}(B)}{\mathrm{vol}(B)}$ is the normalized association.

In [89], the authors show that we can get the good partitions of $G$ when the normalized cut value is minimum. At the same time, it also maximizes the normalized association because $Ncut(A, B) + Nassoc(A, B) = 2$. Back to the text segmentation, the above criterion also maximizes the similarity inside a segment, and minimizes the similarity between different segments. Therefore, it makes a good segmentation. Above normalized cut criterion is easily extended to the general case when the number of segments is $k > 2$.

$$\mathrm{Ncut}_k(V) = \frac{\mathrm{cut}(A_1, V \backslash A_1)}{\mathrm{vol}(A_1)} + \frac{\mathrm{cut}(A_2, V \backslash A_2)}{\mathrm{vol}(A_2)} + \cdots + \frac{\mathrm{cut}(A_k, V \backslash A_k)}{\mathrm{vol}(A_k)} \tag{4.7}$$

The problem of minimizing the normalized cut on graphs is NP-complete, which is proven in [89]. However, in the linear text segmentation, above criterion is constrained to preserve the linearity of the segmentation. It means that all the nodes between the leftmost and the rightmost nodes of a particular partition must belong to the same partition. Malioutov et al. [59] exploited this constrain to develop a dynamic programming algorithm to find the extact solution.

Table 4.2: Systems are involved in experiments

| **Name** | **Description** | **Ref.** |
|---|---|---|
| JTextTile | The Choi's implementation of TextTiling method. | [42] |
| C99 | A widely referred text segmentation system. | [22] |
| TextSeg | A generative-based text segmentation system. | [96] |
| MinCutSeg | A minimum cut segmentation system for spoken lectures. | [59] |
| JSeg | Our system without non-systematic semantic relation. | |
| JSegT | Our system with non-systematic semantic relation ($\lambda = 0.7$). | |

We denote $C[i, k]$ as the normalized cut value of the optimal segmentation of the first $k$ sentences into $i$ segments. The $i$-th segment, $A_{j,k}$ begins at node $u_j$ and ends at node $u_k$. $B[i, k]$ is the back-pointer table that is used to recover the optimal sequence of segment boundaries. We have the following dynamic programming equation:

$$C[i,k] = \min_{j<k} \left[ C[i-1,j] + \frac{\mathrm{cut}(A_{j,k}, V \backslash A_{j,k})}{\mathrm{vol}(A_{j,k})} \right] \tag{4.8}$$

$$B[i,k] = \arg\min_{j<k} \left[ C[i-1,j] + \frac{\mathrm{cut}(A_{j,k}, V \backslash A_{j,k})}{\mathrm{vol}(A_{j,k})} \right] \tag{4.9}$$

Initial values: $C[0, 1] = 0, C[0, k] = \infty, 1 < k < N$ and $B[0, k] = 1, 1 \leq k \leq N$.

The time complexity of the above algorithm is $O(K^2N)$, where $K$ is the number of segments and $N$ is the number of sentences.

## 4.4 Experiments

In this section, we present experiments on the public dataset with the current available systems on text segmentation. Table 4.2 shows the list of systems that are involved in our experiments. Those systems are used with their default parameters. Some systems such as MinCutSeg and JSeg have been optimized on a separate development set which extracted from the experimental dataset. Documents in the development set are not used in experiments.

### 4.4.1 Dataset

The dataset used in this research is CHOI dataset [22], which has been widely used for benchmarking the performance of text segmentation algorithms [22, 46, 96, 59, 67]. This dataset contains 700 documents. Each document is a concatenation of ten text segments. Each segment, in turn, is the first $n$ sentences of a randomly selected document from the Brown corpus[2]. Each document is characterized by the range of $n$. The corpus was generated automatically according to the description in [22]. Table 4.3 shows the dataset statistics.

Table 4.3: CHOI dataset

| Range of $n$ | 3–11 | 3–5 | 6–8 | 9–11 |
|---|---|---|---|---|
| Number of documents | 400 | 100 | 100 | 100 |

The topic model has been estimated on the WIKI dataset, which contains 3,071,253 articles with 6,332,406 distinct words. The vocabulary used for this research contains 233,851 words. The model contains 1,000 topics has been estimated in 200 iterations by GibbsLDA++[3]. The topic inference has been done on all documents in the dataset.

### 4.4.2 Results and Evaluation

The experimental results have been evaluated using two popular metrics $\mathrm{P}_k$ [8] and WindowDiff [82]. Since $\mathrm{P}_k$ and WindowDiff are error metrics, the low value indicates high accuracy. Table 4.4 shows the evaluation of experimental results of all the systems on the Choi's dataset. The $\mathrm{P}_k$ and WindowDiff (WD) scores have been averaged on all the documents in whole dataset and sub-datasets.

The parameter $\alpha$ of JSegT has been set to 0.5 by tuned on the development set containing five documents.

Table 4.4: Experimental results on CHOI dataset

| **System** | **3–11** | | **3–5** | | **6–8** | | **9–11** | | **Average** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathrm{P}_k$ | WD | $\mathrm{P}_k$ | WD | $\mathrm{P}_k$ | WD | $\mathrm{P}_k$ | WD | $\mathrm{P}_k$ | WD |
| JTextTile | 0.524 | 0.649 | 0.473 | 0.541 | 0.513 | 0.635 | 0.533 | 0.739 | 0.516 | 0.644 |
| C99 | 0.143 | 0.144 | 0.115 | 0.115 | 0.104 | 0.104 | 0.112 | 0.112 | 0.129 | 0.130 |
| TextSeg | 0.106 | 0.107 | 0.074 | 0.075 | 0.052 | 0.053 | <u>0.037</u> | <u>0.037</u> | 0.084 | 0.084 |
| MinCutSeg | 0.243 | 0.251 | 0.340 | 0.350 | 0.241 | 0.244 | 0.174 | 0.175 | 0.247 | 0.253 |
| JSeg | 0.129 | 0.130 | 0.091 | 0.091 | 0.107 | 0.107 | 0.121 | 0.126 | 0.119 | 0.121 |
| JSegT | <u>0.035</u> | <u>0.036</u> | <u>0.020</u> | <u>0.020</u> | <u>0.030</u> | <u>0.030</u> | 0.046 | 0.046 | <u>0.034</u> | <u>0.034</u> |

[2]Brown corpus can be freely accessed via NLTK: `http://www.nltk.org`

[3]`http://gibbslda.sourceforge.net/`

### 4.4.3 Discussion

The text segmentation module is normally a part of an application. Therefore, evaluation of the performance of text segmentation systems is difficult and depends on application. In this research, we used a widely-used artificial corpus to evaluate our system, and it may be appropriate for comparing relative performance among text segmentation systems without applications.

The important point to notice in Table 4.4 is that JSegT system got the better result than other systems on the whole dataset. Thus, it confirms that the use of non-systematic semantic relation could help to improve the performance of the text segmentation. In the other hand, the JSeg system, which employs the minimum cut segmentation from MinCutSeg system, got much better result than MinCutSeg in spite of the MinCutSeg has been optimized on the same development set as JSeg and JSegT. The reason may be that the MinCutSeg has been developed for long document such as transcripts of spoken lectures in MIT. Our systems, JSeg and JSegT, use an advanced smoothing technique that is effective for short document.

In the scope of this paper, we also did experiments with the TextSeg system [96], which is a representative of generative methods in text segmentation. TextSeg got stable and good results in experiments, especially in the dataset 9–11. The TextSeg's approach is very different from the similarity-based approach used in this research. Therefore, if the non-systematic semantic relation could be integrated into TextSeg, it may get the potential results.

## 4.5 Related Work

Our approach takes into account the non-systematic semantic relation to improve the performance of text segmentation. This is a potential approach because it is a step to take into account all the lexical cohesion in determining text structure. Morris and Hirst [68] first suggested using the thesaurus to take into account the collocation relation. However, they met the difficult about lacking electronic resource at that time. Some following works such as [6] also tried to use WordNet and thesaurus for text segmentation. Those resources could help the model recognize the systematic semantic relation. Choi et al. [23] used LSA as a method to reduce the number of dimensions of document.

In the other hand, some previous works tried to use topic information in text segmentation. Ferret [35] tried to discover topics from the document itself without any priori knowledge. His approach is only appropriate with very long documents, which contain enough information for building co-occurrence matrix. Eisenstein and Barzilay [34] built a LDA-based model for long documents, which generalizes the generative method invented by Utiyama and Ishihara [96]. Misra et al. [67] follows the generative approach, in which the topic model is used to compute topic distribution for every candidate segment to determine whether that segmentation is optimal. It is different from our approach, in which the topic information is used to directly capture the non-systematic semantic relation.

## 4.6 Summary

We have proposed a method to recognize the non-systematic semantic relation to improve the performance of the text segmentation algorithm. This relation has been integrated in the similarity computation process, which will directly affect the quality of the segmentation. The topic model used in this research has been estimated from a large, topic-balanced corpora, WIKI, which could help the text segmentation model to apply to the wide range of texts. The experimental results have been shown that our system is better than the availabe systems on the Choi's dataset. In the future work, we are planning to incorporate the systematic semantic relation and evaluate our method on the real corpora.

# Chapter 5

# Generating a Hierarchical Structure of Topic-information

This chapter presents our works on generating a hierarchical structure of topic-information (HST) for a readily available hierarchical structure of segments of texts. We firstly give a short introduction to this task. We then present the Perceptron-based learning model for generating a title for a segment of text, and the ranking-based learning model that accounts for the relations between titles when the generated titles are combined to form a HST. Next, we propose the idea of using supportive knowledge in generating a HST, and how to exploit supportive knowledge in the learning models. To examine the effects of our proposed solutions, we do experiments on a public standard corpus and make some comparison with the current state-of-the-art method.

## 5.1 Introduction

A hierarchical structure of topic-information (HST) is a kind of tree structures that looks like a table-of-contents that is a commonly available in the beginning of a book or a long document. In the text summarization field, a HST could be seen as a kind of indicative summary, which is especially suited for locating information in a long document, or a set of documents. For instance, a HST could play a role as a navigation tool for accessing information in a long document on a mobile device [79], or understanding a long, unstructured transcript of an academic lecture, or a meeting [17]. A reader could use a HST to locate all the parts in a set of documents that are relevant to his/her interests. In addition to the above functions, a HST with its meaningful titles and its hierarchical structure of information could help a reader get an overview of the entire contents very quickly.

Given a long document, or more generally, multi-document having the same topic, our goal is to generate a tree, wherein a node represents a meaningful title that summarizes the content of a segment (in a long document), or segments that have similar content (in multi-document). That tree—a hierarchical structure—can be seen as a HST [17]. As presented in Chapter 1, the process of generating a HST for multi-document involves three successive tasks.

**Text segmentation.** Every document is spitted into topically coherent segments of texts.

**Segment combination.** All the segments are merged and combined to form a hierarchical structure (a tree of segments), in which, a node contains segments that contain similar information.

**Title generation.** A title is generated for each node in the above tree. A title is a short text that reflects the content of segments belonging to that node.

In this chapter, we focus on the third task with the assumption that the hierarchical structure of segments already available. In this task, a generating model should not only generating a meaningful title for each segment, but also taking into account the relations between titles. For example, the adjacent titles should not be duplicated, or the title of section and the title of its sub-section should have hierarchical relations in the domain of the content.

So far, the literature on the title generation task is relatively sparse [4, 17, 73]. Angheluta et al. [4] presents an unsupervised approach to generating a title for a segment. The title is the best noun phrase extracted from the segment. Their method uses some simple grammatical rules to identify the noun phrases, and uses term frequencies to score the noun phrases. This approach usually produces well-formed titles. However, the extracted titles are usually too short, with very low quality in reflecting the meaning of the segments. Furthermore, every title is made independently, and therefore, the generated table-of-contents lacks coherence.

Branavan et al. [17] presents a supervised learning model for generating a table-of-contents. Their model intends to address most of the problems of the previous approaches. Their model captures most important features at the word level and the word sequence level in generating titles, such as position of a word, its TF*IDF, part-of-speech information, language model score, and so on. They also propose an additional model, which captures the relations between titles, to generate a more coherent table-of-contents. However, in their experimental results, in spite of better scores in comparison to other baseline models, a large number of titles in the generated table-of-contents are meaningless or not related to the content of the corresponding segments.

In this research, we try to generate a table-of-contents with meaningful titles with the following idea: *a good title should have a topic relation to the text.* Therefore, we should use as much semantic or topic information as possible to help the model to generate a meaningful title that has strong relation to the content. The semantic and topic information, which are supportive knowledge, used in this research are derived from a hierarchical clustering of words [18], and a topic model [13]. We follow a two-stage semi-supervised approach to use supportive knowledge in a discriminative learning model, which is a perceptron-based learning model [66, 54, 53]. The supportive knowledge is derived from un-annotated texts using unsupervised learning algorithms, which are Brown clustering algorithm [18] and Latent Dirichlet Allocation [13].

Our experimental results on the public dataset show that our approach could help the model to produce a table-of-contents with well-formed and meaningful titles. The evaluation results also show that the titles generated by our model have higher quality

- Finding a dynamic programming
    - Shortest paths in the Floyd Warshall
    - Running time of the Floyd Warshall algorithm
    - Constructing a recursive algorithm
- Bellman Ford algorithm for the running
    - Equation for a given function
    - Bellman Ford algorithm from section

Figure 5.1: A portion of a table-of-contents generated by our model.

than those of the baseline model [17]. Figure 5.1 shows a portion of a table-of-contents generated by our model.

## 5.2 Learning Models

In this chapter, we mainly focus on the third task: generating a HST from a hierarchical structure of segments. In other words, we try to generate a tree of titles from a readily tree of segments. To formalize this task, we employ the approach used in [17].

This task is formalized as a structured learning problem for generating a tree of titles $T$ from a tree of segments $S$. The model, in turn, is decomposed into two components:

- *Title generation model* is an incremental Perceptron-based learning model [26]. This model is used to generate a title $t \in T$ for each segment $s \in S$. This model is also a kind of $k$-best model, which can generate $k$ top-ranked titles for each segment $s$.
- *HST generation model* is a ranking-based learning model that is mainly used for capturing the relations between the candidate titles generated by the local model to form a coherent HST.

We begin the presentation of the models with some common notations:

- $S$ is a tree of segments and $T$ is a corresponding tree of titles. Pairs $(S, T)$ are provided as training data.
- Every segment $s \in S$ has a corresponding title $t \in T$ to form a pair (*segment*, *title*).
- All the pairs $D = (s, t)$ are provided as the training data for the title generation model.
- $|D|$ is the size of a dataset $D$.
- $|t|$ is the length of a title $t$.
- $f(s, z)$ is feature vector of a segment $s$ and a partial title $z$.
- $w_l$ and $w_g$ are the weight vectors of the title generation model and the HST generation model, respectively.

### 5.2.1 Title Generation Model

The title generation model aims to generate a list of candidate titles given a segment of text. As presented in Section 2.2, there are three main approaches in title generation: *key phrase extraction*, *sentence compression*, and *statistical generation.* The model used in this research follows the statistical generation approach. However, in stead of using the two-phase strategy, this model incorporates the word selection phase and the title word ordering phase into a linear discriminative learning model. The information used in word selection phase and title word ordering phase is encoded as features in the model.

#### Training Step

**Algorithm 5.1:** Training algorithm for the local model

```
Input:
  - $D = \{(s,t)\}$ is a set of (segment, title);
  - $N$ is the number of iterations;
  - $k$ is the beam size.
Output:
  - $w_l$ is the weight vector of the local model.
1  for $i = 1 \to N$ do
2      forall the $(s,t) \in D$ do
3          for $j = 1 \to |t|$ do
4              $B \leftarrow \text{GetTop}(\text{PartialGen}(B, s), k)$
5              if $t[1..j] \notin B$ then
6                  $w_l \leftarrow w_l + f(s, t[1..j]) - \frac{\sum_{z \in B} f(s,z)}{|B|}$
7                  $B \leftarrow \{t[1..j]\}$
8              end
9          end
10     end
11 end
12 $w_l \leftarrow w_l/(N * |D|)$
```

In the training step, a vine-growth strategy [28] is employed to learn a model for generating a title of a segment of text by an incremental perceptron-based algorithm. The training process simulates the process of building a title $t$ incrementally by appending words inside the given segment s at each iteration (as in Algorithm 5.1). By following this strategy, the size of the search space is exponential to the length of the desired title, therefore, a beam search algorithm is used to reduce that effect. At each iteration, the beam $B$ keeps up to the $k$ most promising partial titles. This strategy has been successfully applied in other natural language processing tasks, such as parsing [26] and chunking [28].

In Algorithm 5.1, $N$ is the number of iterations of the perceptron-based learning algorithm. At each iteration, by using function **PartialGen**, $B$ is grown by appending every word in $s$ to each partial title in $B$ to make a list of partial titles of length $j$. After that, by using function **GetTop**, $B$ is pruned to contain $k$ top-ranked titles based on the score $w_l \cdot f(s,z), \forall z \in B$. In this score, $w_l$ used in iteration $i$ is the weight vector of the iteration $i-1$. If $B$ does not contain the prefix $t[1..j]$ of the title $t$, $B$ is pruned to contain only $t[1..j]$.

**Decoding Step**

**Algorithm 5.2:** Generating a list of candidate titles for a text segment.

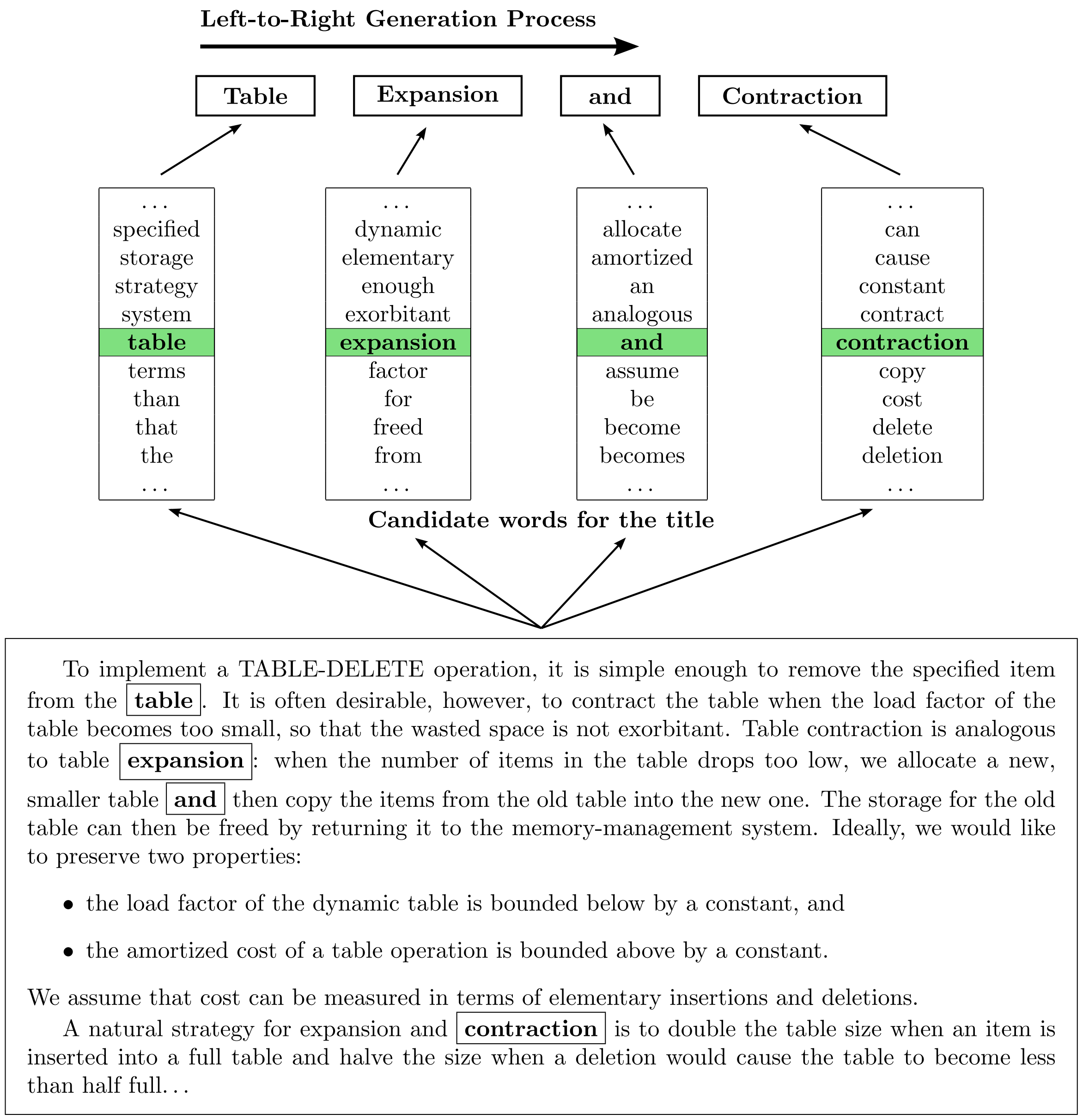
```
Input:
  - $w_l$ is the weight vector of the local model;
  - $s$ is the set of words inside the segment of text;
  - $l$ is the length of the desired title;
  - $k$ is the beam size.
Output:
  - A list of $k$ candidate titles.
1  $B = \{$a set containing an empty string$\}$
2  for $i = 1 \to l$ do
3      $Q \leftarrow \emptyset$
4      forall the $z \in B$ do
5          forall the $w \in s$ do
6              $Q \leftarrow Q + \{z + w\}$
7          end
8      end
9      $Q$ is sorted by $w_l \cdot f(s,z), \forall z \in Q$
10     $B \leftarrow \{$top $k$ partial titles of $Q\}$
11 end
```

In the decoding step of the local model, Algorithm 5.2 produces a list of candidate titles by incrementally generating titles from the words inside the segment of text. The length of the desired title is provided as a parameter of the algorithm [17]. This algorithm uses the same strategy as in training to reduce the size of the search space.

Figure 5.2 shows an example of the title generation process. The title "*table expansion and contraction*" are generated word-by-word from left to right. The list of candidate words for each position in the desired title are the same. It contains all distinct words in the given segment of text.

### 5.2.2 HST Generation Model

Each title in a HST can be generated independently. In other words, we can use the best title among candidate titles generated by the title generation model to form a title of a node in a HST. However, because of the overlap of concepts and topics between adjacent segments, the title generation model may generate duplicated titles. Furthermore, due to the hierarchical structure of a HST, the title of a section should be more general than the titles of its sub-sections. To account for those problems, a higher level generation model


Left-to-Right Generation Process
Table
Expansion
and
Contraction
... specified storage strategy system table terms than that the ...
... dynamic elementary enough exorbitant expansion factor for freed from ...
... allocate amortized an analogous and assume be become becomes ...
... can cause constant contract contraction copy cost delete deletion ...
Candidate words for the title

To implement a TABLE-DELETE operation, it is simple enough to remove the specified item from the **table**. It is often desirable, however, to contract the table when the load factor of the table becomes too small, so that the wasted space is not exorbitant. Table contraction is analogous to table **expansion**: when the number of items in the table drops too low, we allocate a new, smaller table **and** then copy the items from the old table into the new one. The storage for the old table can then be freed by returning it to the memory-management system. Ideally, we would like to preserve two properties:

- the load factor of the dynamic table is bounded below by a constant, and
- the amortized cost of a table operation is bounded above by a constant.

We assume that cost can be measured in terms of elementary insertions and deletions.

A natural strategy for expansion and **contraction** is to double the table size when an item is inserted into a full table and halve the size when a deletion would cause the table to become less than half full...

Figure 5.2: An illustration of the title generation process.

is used to capture the relations between titles to build a coherent HST. We called it the HST generation model.

In the HST generation model, the input is a tree $T$, wherein a node contains a list of $k$ candidate titles of the corresponding segment $s \in S$. The output of this model is a tree with the same structure, wherein a node contains only one title. In this model, the input and the output are hierarchical structures (trees). This is different from the title generation model, in which the input and the output are sequences of words, which are a segment of text and a title, respectively. However, we can still employ the learning and decoding algorithms used in the local model. The technique used here is to traverse the tree of titles in pre-order—the order of titles will appear in the HST. By using this technique, we can also incrementally build the output tree using a similar algorithm as in the local model. The differences here are the elements used in extending and pruning the beam. In the global model, at a node in the pre-order traversing, a beam, which contains a list of $K$ partial trees, is grown by appending every candidate title of that node. After that, the beam is pruned to contain a list of the $K$ top-ranked partial trees. Similar to the local model, partial trees are ranked by score, which is the output value of the perceptron-based model.

The output of the decoding algorithm of the global model is a tree of titles, which is also the desired HST.

Figure 5.3 shows an example of the HST generation process. By traversing the tree of segments in the pre-order mode, titles in the destination HST occur in a linear structure. It is very similar to the title generation process in Figure 5.2, but candidate items for each position are different. Furthermore, at each position in the HST, the algorithm has to take into account the hierarchical information of that position. For example, whether the previous title is in the same sub-tree or is the parent title. Such features are described in Section 5.4.

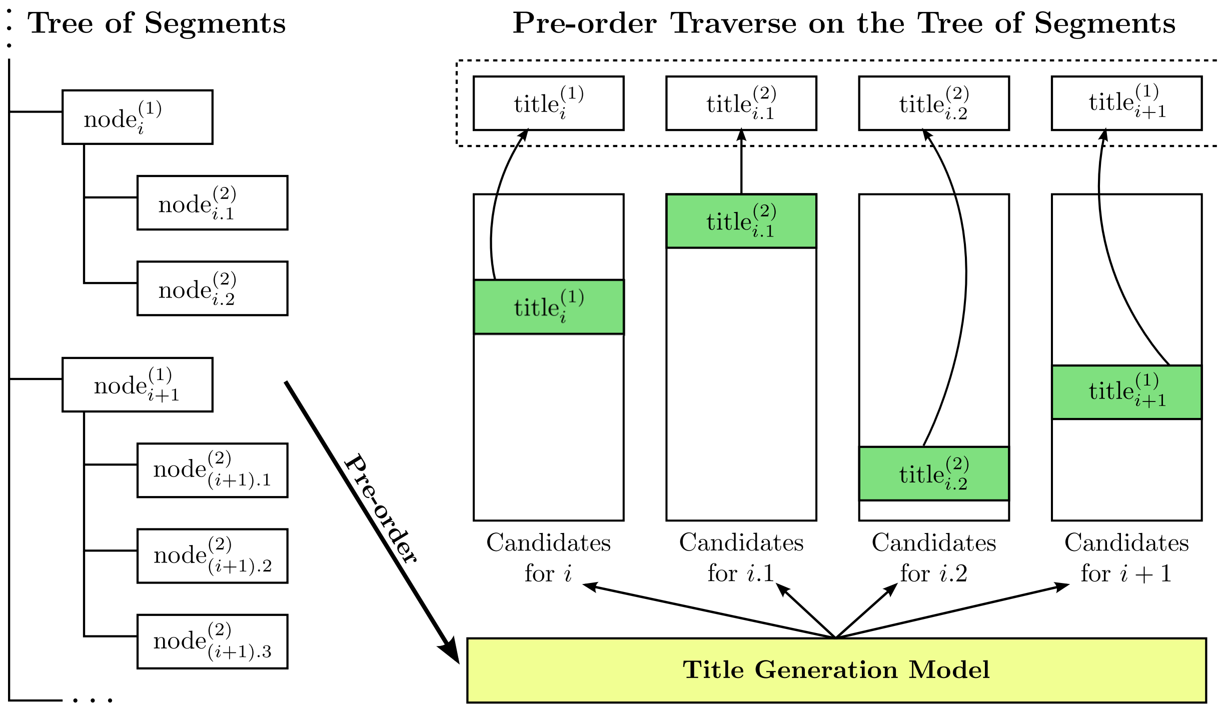

Tree of Segments
Pre-order Traverse on the Tree of Segments
$\text{node}_i^{(1)}$
$\text{node}_{i.1}^{(2)}$
$\text{node}_{i.2}^{(2)}$
$\text{node}_{i+1}^{(1)}$
$\text{node}_{(i+1).1}^{(2)}$
$\text{node}_{(i+1).2}^{(2)}$
$\text{node}_{(i+1).3}^{(2)}$
...
Pre-order
$\text{title}_i^{(1)}$
$\text{title}_{i.1}^{(2)}$
$\text{title}_{i.2}^{(2)}$
$\text{title}_{i+1}^{(1)}$
Candidates for $i$
Candidates for $i.1$
Candidates for $i.2$
Candidates for $i+1$
Title Generation Model

Figure 5.3: An illustration of the HST generation process.

## 5.3 Supportive Knowledge

In this research, we aim to use supportive knowledge to make the learning model take into account semantic and topic information. Supportive knowledge could help the model to improve the quality of the generated titles, and therefore, the generated HST. To reduce the cost of the training process, the supportive knowledge should be easily derived from un-annotated text in an unsupervised way. After that, we incorporate that knowledge as features in a supervised learning model, which is described in Section 5.2. This approach is called a two-stage semi-supervised approach, which was previously used in [66, 54, 53].

To achieve this goal, we firstly try to use word classes, which have two advantages in our task [20]. First, because the classes are created for use in a language model, they should be useful in predicting subsequent words. This is useful in our incremental title generation model. However, at the same time, it is natural to think of such classes as semantic in nature. This could reduce the effects of the data sparsity. Some previous works using a class-based approach treated them in this fashion and got successfully, such as named entity recognition [66], word segmentation [54], and dependency parsing [53]. To get word classes, which could help the model to exploit both the above advantages, the word clustering algorithm should use a similarity measure based on contextual properties of words. In this research, we use the Brown clustering algorithm [18], which has been detailed in Chapter 3. The Brown algorithm gets a large collection of raw text as an input and produce a hierarchical clustering of words. This hierarchical structure allows us to choose an arbitrary number of clusters with an arbitrary level of abstraction. This is the main advantage of this algorithm in comparison to K-means based algorithms.

Secondly, we try to use topic modeling, which could provide topic information in title generation. This is based on the idea: *a good title should reflect most important topics mentioned in the given segment of text.* This relation could be modeled by topic overlapping between the title and the text. In other words, a topic-based similarity measure is used to rank the candidate titles of the text. Thereby, our model could choose the title that best reflects topic information in the text. For instance, in experiments, our model ranks the candidate title "*computing the minimum cost*" (score: $-4.39$, $1^{st}$ rank) in a higher position than "*computing the matrix product*" (score: $-7.69$, position: $4^{th}$ rank) in a segment with the reference title "*computing the optimal costs*". In this research, we choose the most popular topic modeling method, Latent Dirichlet Allocation [13, 39], to estimate and infer the topics from the data. This method has been detailed in Chapter 3.

Both word clustering and topic modeling are used to produce clusters of words from a large collection of raw text. However, word clustering normally produces hard clusters, in which a word can belong to only one cluster. It is different from topic modeling, in which a word can belong to many clusters with different probabilities.

## 5.4 Feature Design

The learning model is decomposed into two components, the title generation model and the HST generation model. Therefore, the feature set is also divided into two subsets for use in those models.

### 5.4.1 Local Features

The goal of the title generation model is to generate a list of candidate titles for a given segment of text. It involves indentifying interest words in the text, and combining them into the title [5]. Therefore, the feature set for this model should capture selection constraints at the word level, and the contextual constraints at the word sequence level. More specifically, the features at word level plays as a filter to select appropriate words to be included in the candidate titles. The features at word sequence level plays as a filter to select and rank the candidate titles via their fluency in language or the relevance between title and the segment of text.

The local features are summarized in Table 5.1. These features are used in the baseline models. As described in Table 5.1, some types of features are normalized. For instance, "Its first occurrence in segment by sentence" is the relative position of the first sentence containing the word, which is normalized by the number of sentences in the given segment.

Table 5.1: Baseline features of the local model for capturing selection constraints at the word level and contextual constraints at the word sequence level.

| **Features** | **Type** |
|---|---|
| **Word level** | |
| Is it a stop word or an auxiliary word? | category |
| Its TF*IDF score | numeric |
| Its part-of-speech | category |
| Its first occurrence in segment by word | numeric |
| Its first occurrence in segment by sentence | numeric |
| Does it occur in the sibling or the parent segments? | category |
| **Word sequence level** | |
| Uni-gram, bi-gram and tri-gram language model scores | numeric |
| The frequency of noun phrases in the word sequence at segment level and corpus level | numeric |

### 5.4.2 Supportive Knowledge Features

The supportive knowledge is incorporated into the local model in the form of features.

For using the word clustering information, each cluster of words is represented by a bit string as described in Section 3.2. To exploit various levels of abstraction of the word clustering, we use corresponding prefixes of the bit string of a word as features. Then, an indicator function is created for each type of prefix and is used as a feature. In experiments, we use three levels of abstraction with three types of the prefix length 4, 6, and 8, respectively. For instance, a indicator function $f^4_{0110}$ can be defined as follows:

$$f^4_{0110}(w) = \begin{cases} 1 & \text{if the 4-bits-prefix of } w \text{ is 0110,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

For the word "language" with bit string "10110111100", the following indicator functions are activated: $f^4_{1011}$, $f^6_{101101}$, and $f^8_{10110111}$. With this representation method, we can limit the number of word clustering features regardless of the cluster size.

To exploit topic information, we use it at both word level and word sequence level. At the word level, the topic distribution information of a word is used. For instance, if the vector of topic counts of a word $w_i$ in a given segment of text is $\vec{z}_i = (|z_i^1|, |z_i^2|, \ldots, |z_i^K|)$, we normalize that vector by the number of occurrences of $w_i$ in that segment. The normalized vector is directly incorporated into the feature set as a selection feature.

At the word sequence level, for each partial title at each iteration of the training and decoding algorithms, the topic distribution is easily computed by normalizing the sum of the vectors of topic counts of all the words in that partial title by the total number of occurrences of all the words in the given segment $s$. For instance, topic distribution of a partial title $t = w_1 w_2 \ldots w_l$ is computed as:

$$p(z_t^i | t, s) = \frac{\sum_{j=1}^{l} |z_j^i|}{\sum_{j=1}^{l} |w_j^i|}, \quad i = 1 \ldots K \tag{5.2}$$

To take into account the relevance between the partial title and the segment of text, we measure the similarity between the topic distribution of the partial title $p_t$ and the topic distribution of the segment of text $p_s$:

$$\mathrm{sim}(p_t, p_s) = 10^{-\beta \mathrm{IRad}(p_t, p_s)} \tag{5.3}$$

where $\beta$ is a scale parameter, which is normally set to 1 in practice, and $\mathrm{IRad}(p, q)$ is the information radius between two distribution $p$ and $q$ [62]. $\mathrm{IRad}(p, q)$, in turn, is defined via Kullback–Leibler divergence $\mathrm{KL}(p, q)$ as follows:

$$\mathrm{IRad}(p, q) = \mathrm{KL}\left(p \middle\| \frac{p+q}{2}\right) + \mathrm{KL}\left(q \middle\| \frac{p+q}{2}\right) \tag{5.4}$$

where

$$\mathrm{KL}(p \| q) = \sum_i p_i \log \frac{p_i}{q_i} \tag{5.5}$$

The above similarity score is incorporated into the feature set as a contextual feature.

### 5.4.3 Global Features

The goal of the HST generation model is to make a coherent HST by choosing the most appropriate title for each node in the tree of titles. This model has to account for the relations between titles in a hierarchical structure. Given a partial tree of titles and a candidate title of the next node in the pre-order traversing, three types of features are used to capture that relation:

- Whether the title is redundant at various levels of the tree: at the sibling nodes, at the parent node.

- The rank of the title provided by the local model via its score. With this feature, the global model can exploit the preferences of the local model in the title generation process.
- The parallel structure of titles that have the same parent node. This phenomena is popular in a table-of-contents, a form of HST. For instance, in the dataset used in this research, a section titled "*Performance of Quicksort*" has three subsections titled "*Worst case <u>partitioning</u>*", "*Best case <u>partitioning</u>*", and "*Balanced <u>partitioning</u>*".

## 5.5 Experiments

### 5.5.1 Data

In experiments, we use a public dataset[1] for training and testing the model [17]. This dataset is, actually, the table-of-contents of the textbook "*Introduction to Algorithms*" [27]. This book contains 564 sections in 39 chapters. The depth of the table-of-contents is 4. The authors of this dataset treated the fragment of table-of-contents of each chapter as a small table-of-contents with depth of 3. Thereby, we have 39 table-of-contents used for training and testing. We divided this dataset into a development set and a test set at a ratio of 80/20. For tuning the parameters of the model, we divided the development set into a training set and a development test set (dev-test set) for training and testing the model before application to the test set. The ratio was also 80/20. Similar to [17], in our experiments, we use ten different randomizations to compensate for the small number of available trees. For each randomization, we have done a 5-fold cross-validation to get the average score.

At the preprocessing step, the dataset is tokenized and tagged by Stanford Log-linear Part-Of-Speech Tagger[2]. The noun phrases are extracted using regular expressions chunking tool in NLTK[3]. SRILM Toolkit[4] is used to train the language model on the training set.

As mentioned in Chapter 3, we use the articles from Wikipedia,the WIKI corpus, to build a hierarchical cluster of words. The Brown algorithm implementation of Liang[5] [54] ran on that corpus to produce 1,000 word clusters. Some clusters are shown in Table 5.2. The number 1,000 is the default setting for large corpora and has been widely used in other research [66, 54, 53]. On the other hand, as described in Section 5.4.2, we only use the prefixes of the bit string representation of the word cluster. Therefore, the number of cluster features is limited regardless of the size of word clustering.

To estimate the topic model, we use the raw data in the training set, in which each section is treated as an independent document. After that, we infer the topic distribution for every section in the training set and the test set. Only the inferred topic information

[1]http://people.csail.mit.edu/branavan/
[2]http://nlp.stanford.edu/software/tagger.shtml
[3]http://www.nltk.org
[4]http://www.speech.sri.com/projects/srilm/
[5]http://www.eecs.berkeley.edu/~pliang/software/

Table 5.2: Sample word clusters derived from WIKI corpus and their bit strings.

| **101010010010** | **101010010000** | **101101101011110** |
|---|---|---|
| sorting | construction | review |
| partitioning | restoration | journal |
| labelling | conversion | selection |
| metering | implementation | survey |
| clustering | execution | encyclopedia |
| formatting | installation | encyclopaedia |
| tunnelling | renovation | timeline |
| ... | ... | ... |

of the training set is used in the learning process as described in Section 5.4. To estimate and infer the topic model, we use the LDA implementation of Newman [70]. In each run, we re-estimate the topic model with 100 topics on the training set. Some topics with their most likely words are shown in Table 5.3. To choose the number of topics, we manually tuned on a development set. Similar to the results in [98], the performance of the model becomes stable when we increase the number of topics, and 100 is a relatively *good* number of topics.

### 5.5.2 Evaluation

The baseline models used in evaluation are the flat discriminative model (Baseline FD) and the hierarchical discriminative model (Baseline HD), which are the best models in [17].

The difference between the flat discriminative (FD) models and the hierarchical discriminative (HD) models is that the hierarchical discriminative models account for global features, which capture the relations between titles in the hierarchical structure. The flat discriminative model omits those relations and simply chooses the highest ranked title from the local model to form the title of a node in the table-of-contents.

We compare the quality of the baseline model to our four models. The first model denoted by *FD+WC* is a flat discriminative model, which uses the local feature set and word clustering based features. The second model denoted by *FD+TM* is a flat discriminative model, which uses the local feature set and topic model features. The third model denoted by *HD+WC* is a hierarchical discriminative model, which is based on the FD+WC model with the global features set. The last model, denoted by *HD+TM*, is a hierarchical discriminative model, which is based on the *FD+TM* model with global feature set.

The experimental results are evaluated using ROUGE metrics [55], which is commonly used to assess the quality of machine-generated headlines [99]. All the scores are averaged

Table 5.3: Most likely words of some sample topics.

| **Topic 1** | **Topic 5** | **Topic 23** | **Topic 81** |
|---|---|---|---|
| circuit | vertex | tree | section |
| input | search | minimum | chapter |
| output | edge | spanning | present |
| combinational | vertices | edge | method |
| element | directed | algorithm | application |
| gate | breadth | weight | basic |
| boolean | discovered | set | finally |
| figure | white | edges | material |
| gates | edges | prim | practical |
| clock | gray | safe | based |
| wire | reachable | trees | examine |
| register | source | section | discusses |
| ... | ... | ... | ... |

over ten randomizations of the dataset. Table 5.4 shows the experimental results with three scores ROUGE-1, ROUGE-L, ROUGE-W, and the number of matched titles, which is the number of generated titles having the same word sequence as original titles.

### 5.5.3 Discussion

Table 5.4 indicates that HD models achieve higher quality than FD models. The reason is that HD models use the global model that captures some useful information about candidate titles, such as the rank of the most matched title generated by the local model, the relations between titles in the same tree (duplication, parallel structure). It is difficult for the local model to take into account that information. On the other hand, when the supportive knowledge is used, the candidate titles are much more relevant to the content of the segment of text. Specifically, 10-best candidate titles contain about 50% matched titles, and 5-best candidate titles contain about 20% matched titles. This means the global model has bigger chance of choosing *good* title. Therefore, the HD models generally have higher quality than the FD models.

As described in Chapter 3, one of the advantages of word clustering is the prediction the next word, which is naturally appropriate to the title generation mechanism used in this research. The experimental results and logs show that it has good effects on choosing words for the candidate titles. Figure 5.4 shows a fragment of a table-of-contents generated by the model HD+WC, in comparison to the same fragment generated by the baseline model, and the reference fragment. In that fragment, HD+WC chose the word

| Reference HST |
|---|
| 11. Hash Tables |
|     11.1. Direct Address Tables |
|     11.2. Hash Tables |
|         11.2.1. Collision Resolution by Chaining |
|         11.2.2. Analysis of Hashing with Chaining |
|     11.3. Open Addressing |
|         11.3.1. Linear Probing |
|         11.3.2. Quadratic Probing |
|         11.3.3. Double Hashing |

| Baseline model's generated HST | Our model's generated HST |
|---|---|
| many dictionaries | dictionary operations |
|     direct address dictionary |     direct address table |
|     computer address |     hash function |
|         chaining a same slot |         chaining all the elements |
|         creating an same chaining hash |         hash table with load factor |
|     address hash |     address hash |
|         hash probing |         hash function |
|         quadratic hash |         quadratic probing |
|         double hash |         double hashing |

Figure 5.4: Fragments of the reference, baseline generated, and our generated HST.

"*dictionary*" followed by "*operations*" to make a title for the segment describing the hash table rather than "*many dictionaries*" by the baseline model. Another advantage of word clustering is the various levels of abstraction of words, which could help the model to reduce the effects of data sparsity. Table 5.4 shows that FD+WC model can achieve higher quality than the baseline model without capturing the relations between titles. On the other hand, word clustering is similar to the part-of-speech in terms of grouping words by functionality. Furthermore, word clustering can group words by the category or topic. Some examples are shown in Table 5.2. These characteristics of the word clustering affects the title generation model in a similar way as the part-of-speech does. In other words, some clusters have the higher impact than others. For instance, the 6-bits-prefix cluster "100100" containing "introduction", "conclusion", "overview"... has a higher weight score than others, because they tend to be occurred in the title more frequently than other words. That is one of the advantages of our approach in comparison to the baseline method. However, it also negatively affects the ranking of candidate titles by promote titles containing common words, especially when the desired length of title is short or the title locate at high level in the table-of-contents. For instance, instead of choosing "recurrences" or some related words as the title of the chapter "Recurrences", the model chose "introduction".

Table 5.4: Results of experiments on public dataset.

| **Models** | **ROUGE-1** | **ROUGE-L** | **ROUGE-W** | **Match** |
|---|---|---|---|---|
| Baseline FD | 0.235 | 0.215 | 0.169 | 10.35 |
| Baseline HD | 0.246 | 0.226 | 0.178 | 11.75 |
| FD+WC | 0.252 | 0.231 | 0.182 | 10.60 |
| HD+WC | 0.301 | 0.290 | 0.229 | 12.80 |
| FD+TM | 0.302 | 0.290 | 0.252 | 13.40 |
| HD+TM | 0.322 | 0.327 | 0.269 | 13.60 |

Table 5.5: Results of experiments that remove some type of feature.

| **Removed features** | **ROUGE-1** | **ROUGE-L** | **ROUGE-W** | **Match** |
|---|---|---|---|---|
| Language Model | 0.167 | 0.143 | 0.106 | 1.10 |
| Position | 0.173 | 0.156 | 0.118 | 6.10 |
| TF-IDF | 0.203 | 0.185 | 0.152 | 8.35 |
| Sibling | 0.219 | 0.200 | 0.156 | 9.10 |
| POS Tag | 0.220 | 0.202 | 0.158 | 9.15 |
| NP Frequency | 0.232 | 0.212 | 0.166 | 10.00 |
| None | 0.235 | 0.215 | 0.169 | 10.35 |

The topic modeling is even better in support of the table-of-contents generation process. Table 5.4 shows the improvement of 0.083 averaged ROUGE-L score in comparison to the baseline model. This is the effect of topic-based similarity score between the title and the given segment of text. By using this similarity score as a feature in a linear learning model, the model could put a title in a higher position if the topic distribution of that title is closer to the topic distribution of the given segment than the others. For instance, in experiments, our model ranked the candidate title "*computing the minimum cost*" (score: $-4.39$, position: $1^{st}$) in a higher position than the other "*computing the matrix product*" (score: $-7.69$, position: $4^{th}$) in a segment with the reference title "*computing the optimal costs*". The reason is that "*matrix product*" is only an example of computing the optimal costs, and therefore, the candidate title containing that phrase is *further* than the first one in terms of topic. However, the use of topic modeling is also noise sensitive. For instance, in a segment of text discusses on a topic with an example of another topic at the end, titles that are close to the main topic usually have low ranks in comparison to general meaning titles. For example, in segment titled "longest common subsequence", the 2-best candidate titles are "representing the problem" and "the dna strands".

To compare the impacts of word clustering and topic modeling on the title generation task with the other feature types in Table 5.1, we did some additional experiments, in which we removed some type of feature from the baseline model. Table 5.5 shows that the most important feature is the language model score with $-0.072$ point of ROUGE-L score. At the word level, the position of a word and its TF-IDF are very important. The impact of POS tag is approximate the impact of the word clustering in Table 5.4, which is suitable to our analysis on the use of word clustering. The impact of frequency of noun phrases is very small and not significant. Table 5.4 and Table 5.5 also show the high impact of topic modeling on the title generation task with $+0.075$ point of ROUGE-L score.

## 5.6 Summary

In this chapter, we proposed a supportive knowledge approach for supporting the HST generation learning model. We also proposed a method of integrating supportive knowledge into the learning model to help the model capture semantic and topic information at both the word level and the word sequence level. The supportive knowledge used in this research is derived from the word clustering, which is acquired from a large collection of raw text by an unsupervised learning algorithm. Another type of supportive knowledge is derived from a topic model, which is directly estimated from the training dataset. We performed experiments on a public dataset and obtained relatively good results in comparison to the current state-of-the-art model.

Our approach is general enough to be applied to other tasks in text summarization that need semantic or topic information, such as summary generation, sentence extraction, or sentence compression.

# Chapter 6

# Generating a Hierarchical Structure of Topic-information for Multi-documents

In this chapter, we present our model on generating a hierarchical structure of topic-information for multiple documents written about the same topic. This work is a combination of previous works presented in Chapters 3, 4, and 5. First, we give an introduction on generating a HST for multi-documents. Next, we present the some differences in using supportive knowledge in multi-document case. We then present our algorithm for combining segments to form a hierarchical structure of segments. Finally, results of some experiments on the real datasets are shown and discussed.

## 6.1 Introduction

A HST of multiple documents written about the same topic is a useful structure which can be used as a navigator for locating the interesting parts or to get an overview of the topic quickly. Our proposed framework for this task has been shown in Figure 1.1. In this study, we intend to use supportive knowledge to provide semantic and topic information to improve the quality of the HST generation model. Specifically, the model automatically acquires supportive knowledge from a large collection of documents such as WIKI corpus—a multi-purpose and multi-domain supportive knowledge. Next, the model splits every document to segments that are topically independent. Those segments write about different aspects of the main topic, which is called sub-topics in this research. The supportive knowledge is used in this step to improve the quality of the text segmentation process by take into account the systematic and non-systematic semantic relation. The model then does segment combination step to make a composite tree of segments that represents the internally hierarchical structure of sub-topics. Finally, in HST generation step, a title is generated for each node in that tree to form a HST. The supportive knowledge used in this step is to help the model generate the title that reflects topic information of the segment.

In this chapter, we focus on the HST generation for multi-documents, which has some differences from the HST generation for single document in Chapter 5. First, the sub-topics are distributed over documents in the set. Thus, the model has to identify them. It helps the reader locate the interesting parts or segments. Second, the model has to organize the sub-topics in some structure to help the reader take an overview of the content of the set. This structure could be flat, in which the segments have been ordered by time, or be hierarchical, in which the segments have been put into a hierarchical structure of topics. Third, a set of documents normally have an internal structure of information that can be exploited to identify the discussed sub-topics. This is an advantage of the multi-document case in comparison to the single document case.

In the next section, the supportive knowledge used in the multi-document case is discussed. Next, we present our model that is used to build a hierarchical structure of segments that reflects the hierarchical structure of topics inside the set of documents. We then present some experiments on the real dataset and discuss about the applicability of our model in the real world.

## 6.2 Supportive Knowledge

In Chapter 3, we mainly focus on acquiring supportive knowledge from a large collection of documents. This method has some advantages. First, the acquired vocabulary is large enough to cover a wide range of categories and topics. Second, the acquired supportive knowledge is close to natural distribution and can be used in a wide range of domains. In other words, it is nearly domain-independent. In the case of topic modeling, it is possible to do topic inference on any new document to get the topic distribution. However, a topic model estimated on a large corpus such as WIKI normally contains very general topics, such as geographic, economics, technology, and so on. Therefore, in case of multi-document written about the same topic, it is difficult for such a topic model to cover smaller topics, such as laptop, monitor, and processor, etc. in topic technology, or gold price, stock market, and real estate market, etc. in topic economics. To estimate a topic model that could cover small topics in multi-documents, we can use an in-domain corpus. For example, we can estimate a topic model on the articles of category "Business" of The New York Times to recognize domain-specified topics in a set of documents written about the fluctuation of the gold price.

Our solution in the multi-document case is similar to what we use in the Chapter 5. Given a set of documents with the same topic, we estimate a topic modeling directly on that set. The estimated topic model, therefore, can cover small topics that are discussed. Therefore, we have two solutions of using supportive knowledge as follows.

- The topic model has been acquired from outside corpus, such as WIKI. Next, every document has been topic-assigned. Then, the steps text segmentation, segment combination, and title generation have been done normally with the support of topic information.
- The second solution is different from the first solution in the intercept point between text segmentation and segment combination. A new topic model has been estimated on the segments, in which a segment is treated as a document. The words in each

segment are then re-assigned using the new topic model. That topic information will be used later in the segment combination and the title generation steps.

A disadvantage of the second solution is that the number of segments is relatively small. Therefore, in many cases, it is not enough co-occurrence information to estimate a "good" topic model. In experiments, we investigate both methods on the real dataset.

## 6.3 Segment Combination

Segment combination is the process of combining segment of texts into a composite tree that reflects the internal hierarchical structure of information in a set of documents written about the same topic. This problem has been raised in our model for the multi-document case. The input for the segment combination step is a collection of segments taken from the output of the text segmentation step. The output is a tree of segments that will be the input of the HST generation step. We propose this step to reduce the efforts on changing the text segmentation and HST generation for the multi-document case.

Our main purpose of this task is to build a tree of segments that reflects the internal hierarchical structure of topic-information. In other words, we try to combine segments that have similar content and topic into a node, and then combine nodes that reflect aspects of the same topic into a bigger node with the higher level of abstraction. Such a description of segment combination step brings us to a clustering solution, which is based on the hierarchical agglomerative clustering method described in Section 2.3.2. Our proposed algorithm is presented in Algorithm 6.1.

**Algorithm 6.1:** A HAC-based algorithm for segment combination.

```
Input: A set of $K$ segments $\{s_1, s_2, \ldots, s_K\}$.
Output: A tree of $K$ segments.
1 $\mathbf{n}^{(1)} = \{s_1, s_2, \ldots, s_K\}$
2 for $k = 1$ to $K-1$ do
3     $\mathbf{sim}^{(k)}$ = pairwise similarity matrix of $n^{(k)}$
4     $(n_i^{(k)}, n_j^{(k)}) = \arg\max_{n_i, n_j \in n^{(k)}} \mathbf{sim}^{(k)}(n_i, n_j)$
5     $c^{(k)}$ = combination of $(n_i^{(k)}, n_j^{(k)})$
6     $\mathbf{n}^{(k+1)} = \{c^{(k)}\} \cup \mathbf{n}^{(k)} \setminus \{n_i^{(k)}, n_j^{(k)}\}$
7 end
```

In the above algorithm, the input is a set of $K$ segments $\{s_1, s_2, \ldots, s_K\}$ represented as $K$ nodes $\{n_1^{(1)}, n_2^{(1)}, \ldots, n_K^{(1)}\}$. The main part of the algorithm is a loop with $K-1$ steps. At each step $k$ $(0 < k < K)$, the algorithm computes the similarity scores between every pair of nodes $\mathbf{sim}^{(k)}$. Next, it chooses the two nodes $n_i^{(k)}$ and $n_j^{(k)}$ that are most related (maximum similarity score) to form a unique node $c^{(k)}$, whose content is a combination of $n_i^{(k)}$ and $n_j^{(k)}$. The remaining $K-(k-1)$ nodes and the new node form the input for the next loop $\mathbf{n}^{(k+1)}$. After $K-1$ steps, the algorithm output a tree of segments, which reflects the hierarchical structure of information.

In Algorithm 6.1, there are two points that have to be defined clearly. The first point is the node representation, which is also used to represent a composite node. The second point is the similarity score function, which is used to compute the degree of relationship between two nodes in terms of content.

### 6.3.1 Node Representation

A node, which is a segment of text, is normally represented by a vocabulary vector, in which each element of the vector is the frequency of the corresponding word in that segment. This representation is very spare due to the small number of words occurred in the text in comparison to the number of words in the vocabulary. A disadvantage of this representation is that it only takes into account the relation between surface words, without meaning or any semantic information. Therefore, it is difficult to recognize the hierarchical structure of sub-topics with this representation.

Similar to the text segmentation step, we intend to use supportive knowledge to provide semantic and topic information for the segment combination step. With the topic information, the relation of two segments is not only based on the word overlapping, but also the topic overlapping. This approach, therefore, could group segments (or nodes) by topics to form a hierarchical structure of topics discussed in the set of documents. Consequently, in our approach, a segment is represented by two vectors: a vector of word frequencies and a vector of topic distribution.

To make a vector of a cluster, we treat the cluster as a large text which is formed by concatenating text from all segments that belong to that cluster. The vector of word frequencies and the vector of topic distribution are re-calculated in the same way as of the single segment.

### 6.3.2 Similarity Score Function

To take into account both types of representation of a node, we use a linear combination of two similarity scores, which are corresponding to two representations, respectively. This method is similar to the way of computing the similarity score between two sentences, which is used in the text segmentation task. Specifically, the similarity score between two nodes $n_i$ and $n_j$ is computed as follows.

$$\text{sim}(n_i, n_j) = \lambda \text{sim}_{lex}(n_i, n_j) + (1 - \lambda)\text{sim}_{topic}(n_i, n_j) \tag{6.1}$$

where $\text{sim}_{lex}(n_i, n_j)$ is the similarity of two vectors of word frequencies, and $\text{sim}_{topic}(n_i, n_j)$ is the similarity score of two topic distributions. The $\text{sim}_{lex}$ is computed by Equation 4.1 and $\text{sim}_{topic}$ is computed by Equantion 4.3. In practice, we choose $\lambda = 0.5$ to make the balance between lexical and topical properties.

### 6.3.3 Flatten the Binary Tree

A disadvantage of Algorithm 6.1 is that the output is a binary tree (dendrogram). In the real world, a node in the tree of segments tends to have more than two children to reflect more than two aspects of a topic. To address this problem, we propose a flattening strategy, which puts adjacent segments that are joined in consecutive steps into a flat structure. For example, Table 6.1 showed the log of the combination process of a 11-nodes set. The first column is 10 steps of Algorithm 6.1. The second and third columns are the chosen nodes to be combined to make a new node in the fourth column. The horizontal lines separate consecutive nodes to be flattened.

Table 6.1: An example of combination steps of Algorithm 6.1 on a set of 11 nodes.

| **Step** | **Node 1** | **Node 2** | **New node** |
|---|---|---|---|
| 1 | $n_1$ | $n_2$ | $n_{12}$ |
| 2 | $n_3$ | $n_{12}$ | $n_{13}$ |
| 3 | $n_4$ | $n_{13}$ | $n_{14}$ |
| 4 | $n_6$ | $n_7$ | $n_{15}$ |
| 5 | $n_5$ | $n_{15}$ | $n_{16}$ |
| 6 | $n_8$ | $n_9$ | $n_{17}$ |
| 7 | $n_{10}$ | $n_{11}$ | $n_{18}$ |
| 8 | $n_{16}$ | $n_{17}$ | $n_{19}$ |
| 9 | $n_{18}$ | $n_{19}$ | $n_{20}$ |
| 10 | $n_{14}$ | $n_{20}$ | $n_{21}$ |

Figure 6.1 illustrates the flattening process of the log showed in Table 6.1. A 5-depth binary tree has been flattened to make a 2-depth tree.

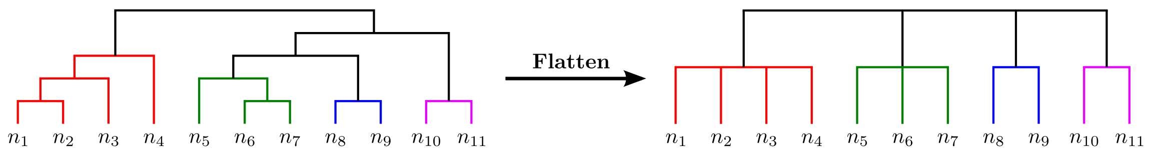

Flatten
$n_1$ $n_2$ $n_3$ $n_4$ $n_5$ $n_6$ $n_7$ $n_8$ $n_9$ $n_{10}$ $n_{11}$
$n_1$ $n_2$ $n_3$ $n_4$ $n_5$ $n_6$ $n_7$ $n_8$ $n_9$ $n_{10}$ $n_{11}$

Figure 6.1: An example of flattened tree.

## 6.4 Experiments

In this chapter, we do two experiments on the real datasets to (1) verify the proposed segment combination algorithm and (2) test our HST generation model for multi-documents, respectively. The first experiment has been done on the dataset ALG, which is used in Chapter 5, since they contain readily hierarchical structures.

### 6.4.1 Segment Combination

In this experiment, we use the ALG dataset as in Chapter 5. The table-of-contents of each chapter with depth of 2 is treated as a set of documents written about the same topic. Specifically, each section is treated as a document, in which each sub-section is treated as a segment. Consequently, we have 39 sets of documents. Each set contains several documents organized in a hierarchical structure.

Figure 6.2 shows some output of experiments. Each row contains two trees, in which the left one is the reference tree, and the right one is the generated tree, which is the output of Algorithm 6.1.

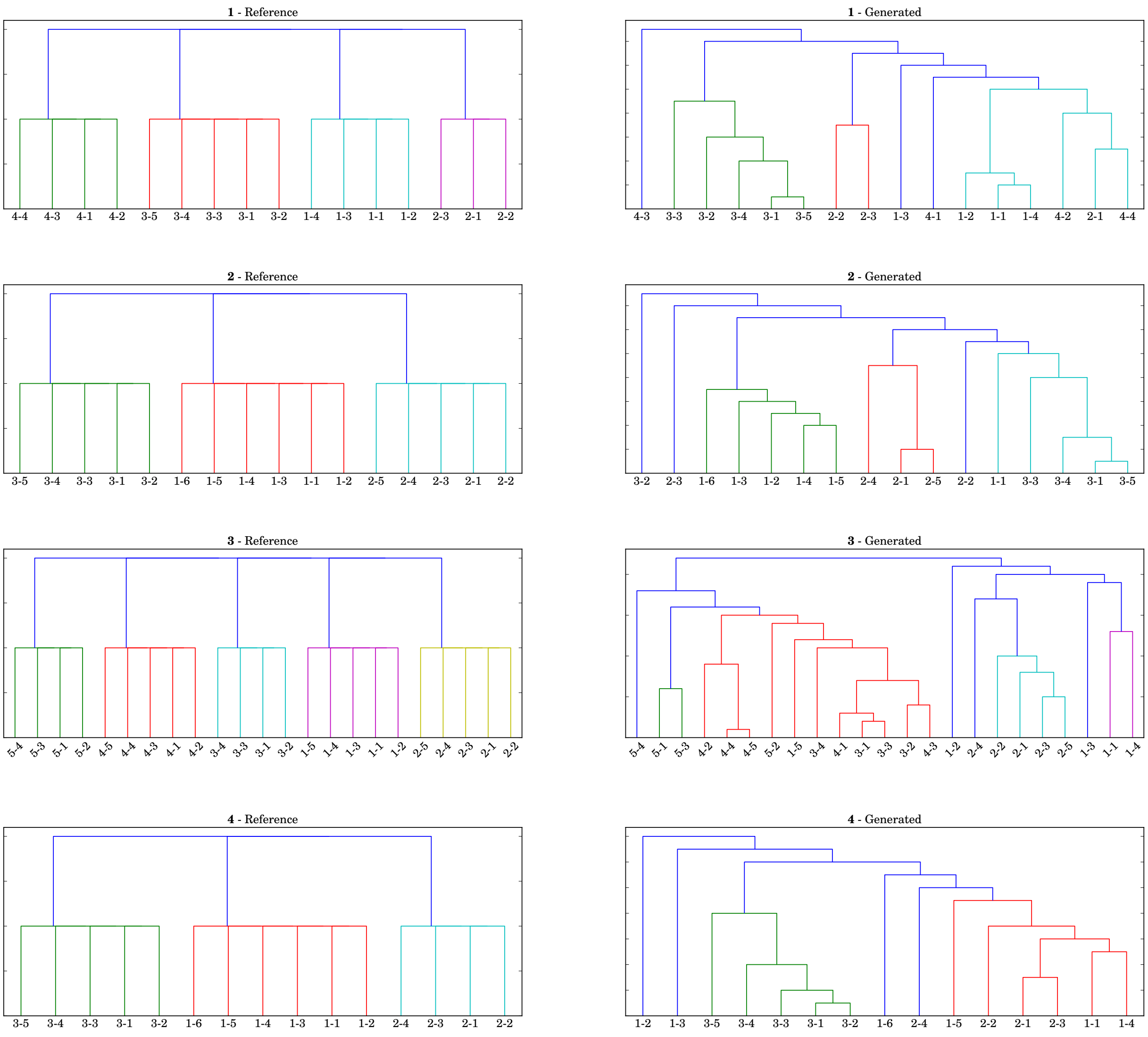


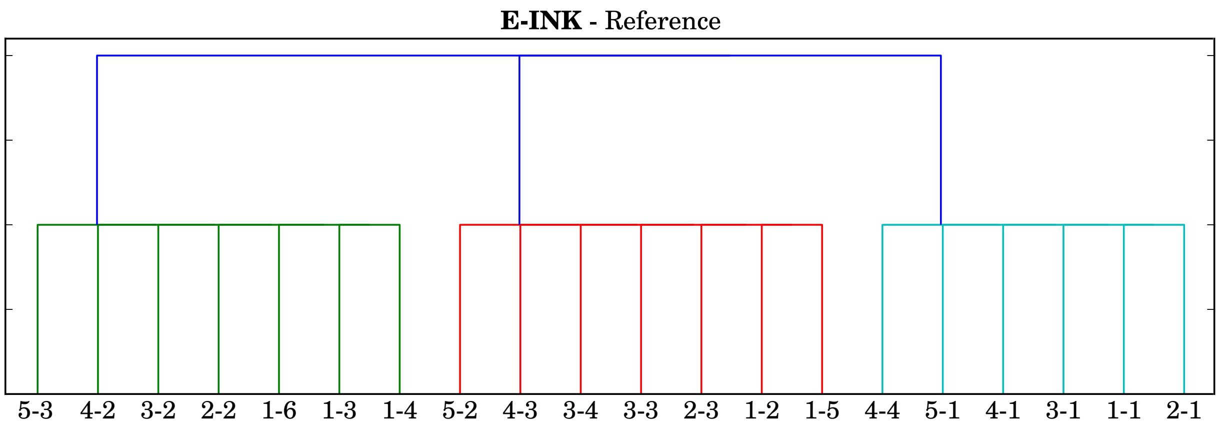
Figure 6.2: Some composite trees of experiments on ALG dataset.

### 6.4.2 Generating HST for Multi-documents

The main purpose of this experiment is to check the ability of our HST generation model in the real world. We use the E-INK dataset that is a set of news articles written about the presentation of the first color E-Ink e-book reader at the FPD International 2010[1] trade show in Tokyo. The detail of dataset is presented in Appendix B.

In this experiment, the E-INK dataset is firstly topic-assigned and segmented automatically. Next, all segments are merged using Algorithm 6.1. Finally, a title is generated for each segment. The length of each title is fixed to be 5 words. The language model is estimated directly on the E-INK dataset. The topic model used in this experiment is estimated on the WIKI dataset with 1,000 topics. The title generation model is trained on the ALG dataset. This may be a strange strategy. However, as presented in Chapter 5, the title generation model is not dependent on words. Furthermore, the title used in a book is normally content-based, which is different from the title used in news articles that are normally attractive-based. Thus, we can apply the trained model in the outside domain.

In Figure 6.3, we show reference the tree of segments, which is created manually, along with the generated tree by our text segmentation and segment combination algorithms. Figure 6.4 shows the flattened version of the above generated tree of segments. The generated HST of the E-INK dataset is shown in Table 6.2.



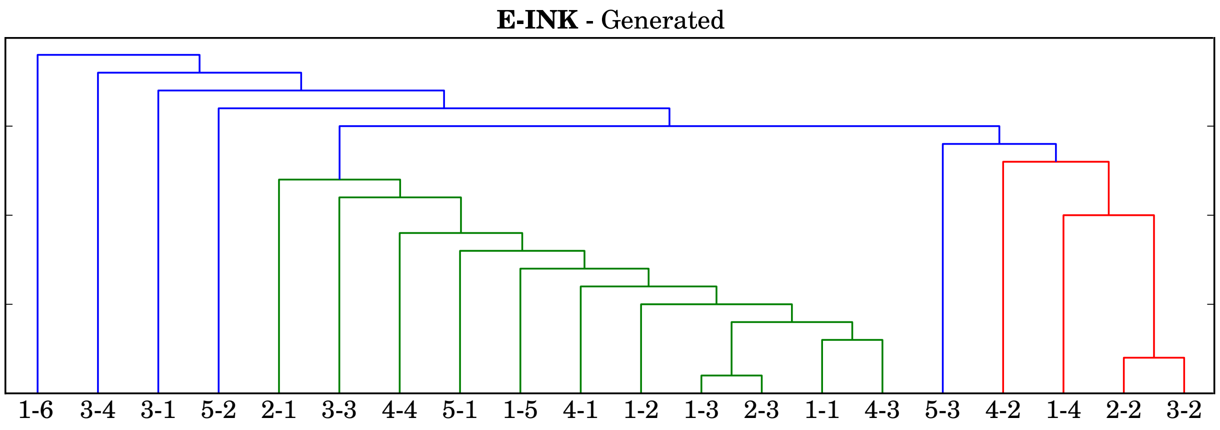


Figure 6.3: The manually created tree of segments and the generated tree of segments of E-INK dataset.

[1]Comprehensive Exhibition and Convention on Flat Panel Displays - FPD International 2011: `http://expo.nikkeibp.co.jp/fpd/2010/english/`

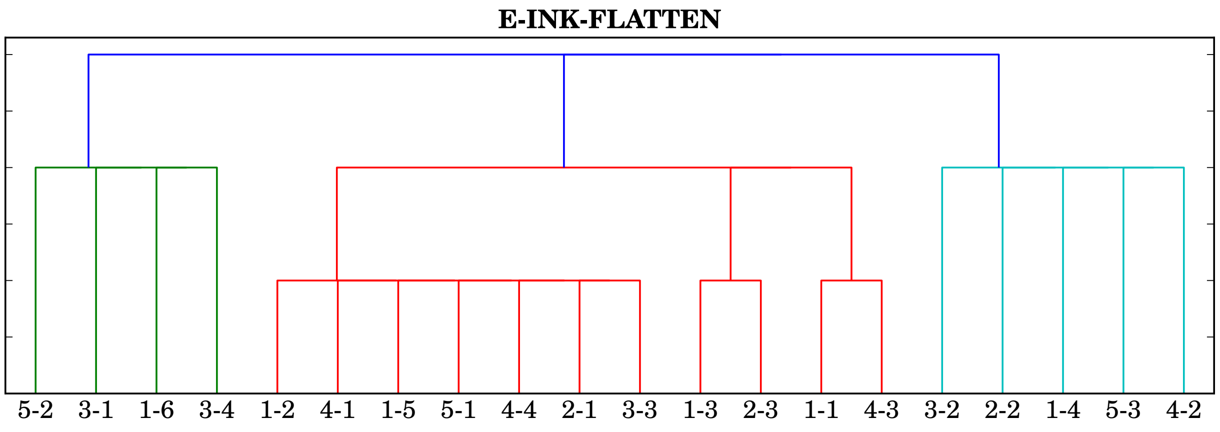


Figure 6.4: The flattened version of the generated tree of segments.

Table 6.2: The generated HST of E-INK dataset using the generated tree of segments.

| Section | Generated title | Segments (location) |
|---|---|---|
| root | havon e-reader early next year | |
| 1 | reading device like the ipad | 1-6, 3-1, 3-4, 5-2 |
| 2 | e ink display color filter | |
| 2.1 | e ink display the e-reader | 1-2, 1-5, 2-1, 3-3, 4-1, 4-4, 5-1 |
| 2.2 | color e ink tech sony | 1-3, 2-3 |
| 2.3 | e ink display color technology | 1-1, 4-3 |
| 3 | next year about 440 | 1-4, 2-2, 3-2, 4-2, 5-3 |

### 6.4.3 Discussion

Figure 6.2 shows that the quality of the generated tree of segments is still very low. The structure of the generated tree is much different from the structure of the reference tree. A positive point is that the segments which have the same parent node are mostly in the same document in the dataset. That means our model has the ability of combining the segments discussed about the same aspect of the main topic.

Table 6.2 shows the HST of E-INK dataset, in which titles are good in terms of readability and meaning. This HST is generated based on the flattened tree of segments in Figure 6.4. However, the tree is not good in reflecting the reference structure of information of the dataset. Therefore, the HST does not reflect enough aspects of the topic. Some aspects that are well reflected in the generated HST are: introduction of a new device like the iPad, the color E Ink technology, the price of color E Ink device, and the time of releasing the color E Ink. It lacks some aspects, such as: the situation of the e-reader market, the plan of distributing color E Ink in China and USA.

## 6.5 Evaluation

So far, we have evaluated the text segmentation step in Chapter 4 and the HST generation step in Chapter 5. However, we have not evaluated the whole system.

Currently, there is no standard corpus for the task HST generation for multi-documents. Therefore, we have to employ two experts who are students with good English skill (TOEFL PBT[2] score $\sim$ 600) to evaluate the output of our system. We call them Judge 1 and Judge 2, respectively. Our system aims to support end-users in news reading. Thus, it is appropriately to use users' satisfaction to evaluate the quality of the system.

The evaluation process is as follows. We firstly collect articles automatically from news cluster services such as Techmeme[3] and Google News[4]. These articles are already grouped by specific topics. Each set of articles belonging to a topic is viewed as a dataset. Next, we run the system on each dataset to get a HST. Finally, experts are requested to score each HST according to a list of criteria. Each criterion score ranges from one to five based on their satisfaction, in which one is for dissatisfaction and five is for full satisfaction.

Table 6.3 shows the list of criteria used in our evaluation with descriptions. These criteria are designed to fit the purpose of the HST in this study.

Table 6.3: Evaluation criteria of HST generation for multi-documents.

| **Abbr.** | **Criterion** | **Description** |
|---|---|---|
| **C** | Coverage | How does the output cover the content? |
| **O** | Order | Does the order of titles fit to the content? |
| **H** | Hierarchy | Is the hierarchical structure reasonable? |
| **R** | Readability | How about the readability of titles? |
| **G** | Grammatically | How about the grammaticality of titles? |

Beside E-Ink dataset used in experiments in previous section, we also run our system on the more four datasets. Table 6.4 show the description of the datasets with the time of events.

In Table 6.5, we summarize the evaluation results from judges. Although the results are not high, it shows the potential application of this study. Judges also give some comments. First, they said that the readability and grammatically are good. Second, they agree that the fixed length of titles is a disadvantage. In practice, deciding the number of words for a title is a difficult task for not only machine but also people. Next, they are very impressed on the topic-information hierarchy, but the order of titles is not good. The reason is that we have not taken into account the information ordering in our system.

[2]Test of English as a Foreign Language (TOEFL): Paper-based Test (PBT)

[3]Techmeme: `http://www.techmeme.com`

[4]Google News: `http://news.google.com`

Table 6.4: Datasets used in evaluation.

| **Time** | **Dataset** | **Description** |
|---|---|---|
| Nov. 8, 2010 | E-Ink | The presentation of the first color e-ink reader. |
| Dec. 1, 2010 | GReader | Google Reader is released for Android platform. |
| Dec. 15, 2010 | Zuckerberg | Mark Zuckerberg is chosen as the person of the year 2010 by Times. |
| Dec. 16, 2010 | AppStore | Apple to release an App Store for Mac. |
| Dec. 17, 2010 | WordLens | Word Lens, an iPhone application, translate words inside of images. |

Finally, the coverage should be improved because it is the most important criterion. The reason of this problem may be that some segments discuss several topics, therefore, only one title is generated for those topics. It is the problem of the text segmentation task.

Table 6.5: Datasets used in evaluation.

| **Dataset** | **Judge 1** | | | | | | **Judge 2** | | | | | | **J-Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | C | O | H | R | G | Avg. | C | O | H | R | G | Avg. | |
| E-Ink | 4 | 2 | 4 | 2 | 4 | 3.2 | 3 | 3 | 2 | 2 | 3 | 2.6 | **2.90** |
| GReader | 4 | 2 | 3 | 3 | 3 | 3.0 | 2 | 2 | 3 | 3 | 3 | 2.6 | **2.80** |
| Zuckerberg | 3 | 3 | 4 | 4 | 4 | 3.6 | 4 | 3 | 3 | 4 | 3 | 3.4 | **3.50** |
| AppStore | 3 | 3 | 2 | 4 | 4 | 3.2 | 3 | 2 | 2 | 3 | 3 | 2.6 | **2.90** |
| WordLens | 4 | 2 | 3 | 4 | 4 | 3.4 | 3 | 3 | 3 | 3 | 2 | 2.8 | **3.10** |

## 6.6 Related Work

In our proposed framework, the hierarchical structure of segments or the tree of segments play an important role. The final HST is built based on that tree. We have proposed a two-step process to build that tree. First, we linearly split every document into topically coherent segments. Then, we make a tree of segments by discovering the hierarchical structure of segments.

Some previous works have been done in building a hierarchical structure of segments. However, most of the works had done on the single document case. Yaari [103] proposed a hierarchical clustering algorithm in combination with lexical cohesion to build a tree of paragraphs. He also extended his work to build a simple table-of-contents for single document [104], in which titles are generated using key phrase extraction method. Slaney and

Ponceleon [91] employed a scale-space segmentation technique from the image processing field to discover the hierarchical structure of segments on the latent semantic indexing (LSI) space. Angheluta et al. [4] applied a linear segmentation algorithm recursively to retrieve the nested structure of segments. She also made a simple table-of-contents in the same way with [104]. Recently, Eisenstein [33] used a Bayesian latent topic model to find a hierarchical structure of segments. Carroll [19] had proposed an evaluation method for hierarchical segmentation algorithm on single document.

The most related work is [40], in which Haghihi and Vanderwende proposed a Bayesian model to discover the hierarchical structure of sentences in a set of documents to choose the representative sentences to be included in the summary. However, in this study, we intend to discover the hierarchical structure of segments, which are topically coherent.

## 6.7 Summary

In this chapter, we have presented our model on generating a HST for multi-documents. We discussed some main differences between the single document case and multi-document case in both text segmentation step and supportive knowledge acquisition. A segment combination algorithm, which is based on HAC, has been proposed along with a flattening method to beyond the limitation on the binary tree of HAC-based methods. Some experiments on the real datasets have been described and discussed.

# Chapter 7

# Conclusions and Future Directions

## 7.1 Summary of the Thesis

In this thesis, we have presented a study on generating a hierarchical structure of topic-information for multiple documents with the focus on using supportive knowledge to improve the quality of the models. The study considers both theoretical and practical views of three tasks in our research problem which are text segmentation, segment combination, and title generation. The thesis consists of six chapters. The first chapter gives an introduction to the research context and the content of the thesis. The second chapter briefly presents background knowledge and previous works of the tasks of this study. The third chapter presents the supportive knowledge. The next three chapters present our works on tasks of HST generation with supportive knowledge. We also give a demostration on a real dataset in Chapter 6 to check the ability of applying our model in the real world.

The main contributions of this study are summarized as follows.

- First, we proposed a model for generating a HST for multiple documents written about the same topic, which is a new problem in natural language processing. Our model is a combination of three tasks those are text segmentation, segment combination, and title generation, in which the segment combination is a new task that is raised in this problem. The experiments on the real dataset show the potential application of this task.
- Second, we investigated the supportive knowledge that is acquired from a large and topic-balanced collection of texts. Supportive knowledge used in this study is a kind of semantic knowledge that can be used to capture the semantic relation between words, sentences, or texts. Two kinds of supportive knowledge used in this study are word clustering and topic modeling. In word clustering, words are grouped by categories or language functions. In topic modeling, words are grouped by topic. Those characteristics play an important role in the tasks investigated in this study. We also built a system for crawling and parsing millions of Wikipedia's articles to make the corpus to derive the supportive knowledge.
- Third, we discovered that the supportive knowledge can be used in the text segmen-

tation task to recognize the systematic and non-systematic semantic relation. They are two most complex relationships in lexical cohesion, which is the main assumption of almost unsupervised text segmentation algorithms. We incorporate such information into a graph-based segmentation algorithm. The experimental results showed that these relationships have positive effects and improve the performance of the text segmentation task in our HST generation model.

- Fourth, we investigate the use of supportive knowledge to improve the performance of the title generation task. The current title generation models only use features taken from surface words, their functions and positional properties. In this study, the semantic and topic information is taken into account to help the model generate titles that reflect the topic of the text. We also proposed a method to incorporate such information into the learning model of title generation step, which exploit the internal structure of word clustering and topic modeling. The experimental results on the public dataset showed that the topic information plays an important role in the title generation and HST generation tasks.

- The final contribution is the proposed HAC-based segment combination algorithm to put a collection of segments into a composite tree. This structure reflects the internal hierarchical structure of information of a set of documents. The drawback of the HAC-based algorithm is that the output is a binary tree, which is not realistic. Therefore, we proposed a flattening method that makes the adjacent segments in the hierarchical structure become flat. We also made some experiments on the real dataset with human evaluation to verify our method.

## 7.2 Future Directions

The main motivation of this study is based on the current demands of people in an information society, who faced with the information overload. Although a large number of news aggregator websites could help people easily get news articles relevant to the same event, they still contain much redundant information and has no structure of information. We intend to build a hierarchical structure of tiles that reflects the information discussed in such a set of news articles. This structure could help the reader quickly get an overview of the event and locate the interesting parts. Based on this motivation, we plan some future directions of this study as follows.

**Research** One of our future works is to pay more attention to remaining issues addressed throughout this thesis. In Chapter 4, we have improved the performance of the text segmentation task with supportive knowledge. Although the experimental results overcome the current state-of-the-art result, our model has no mechanism for determining the number of segments automatically. Therefore, we have to investigate a theoretical and practical analysis to propose a criterion to determine the number of segments.

We also have to do more works on title generation to improve the readability, fluency of the generated title. We will also investigate the way to determine the length of generated title automatically.

In the segment combination task, a challenge is how to build a tree of segments that reflects the hierarchical structure of information. We will investigate on some generative model such as hLDA [9] to deal with this obstacle.

**Application** We plan to implement a module that can retrieve a set of documents as the input and produce the hierarchical structure of topic-information. That system contains three modules corresponding to three tasks: text segmentation, segment combination, and title generation, respectively. With that system, we can easily implement our improvements and verify them on the real data. Furthermore, it can be easily integrated into the readily news aggregator to provide an option to the users to quickly access needed information. We hope this application is a useful and attractive part of a news aggregator or newspaper website.

# Appendix A

# Cohesion in English

Halliday and Hassan [41] describe *texture* as a property possessed by a text, but which an arbitrary combination of sentences does not have. Readers can frequently tell whether or not a series of sentences exhibits texture. In the following example, the sentences in (a) do exhibit it, while those in (b) do not [41].

(a) *Wash and core six cooking apples. Put them into a fire proof dish.*

(b) *Wash and core six cooking apples. The prices of computers drop regularly.*

*Cohesion* is one of the elements of a discourse which contributes to its texture. Cohesion is present when an element in a text is best interpreted in light of a previous (or less frequently, following) element of the same text. Halliday and Hasan identify five cohesive relations which contribute texture to a document. The details are summarized by Reynar [87] as follows.

**Reference** are like pointers. Rather than repeat a phrase in the text, a writer or speaker may use a pointer to the entity selected by a phrase instead. Halliday and Hasan distinguish two main types of reference. *Exophoric references* are to entities in the world of the discourse and *endophoric references* are to portions of the text itself. The word "he" in (a) is an exophoric reference and so is an endophoric reference in (b).

(a) *John likes apples, but he loves pears.*

(b) *For he's a jolly good fellow. And so say all of us.*

**Substitution** and reference are similar, but differ in that substitution occurs prior to semantic interpretation while reference occurs after interpretation. That is a substitute acts merely as a pointer to a region of text which refers to an entity in the world of the discourse, while a reference refers directly to an entity without the mediation of the original referring phrase. In the following example, "does" substitutes for the phrase "like apples": *Do you like apples? Everybody does.*

**Ellipsis** is similar to substitution. It can be viewed as substitution by a zero. In the following example, "bought" has been replaced by a null phrase in "Mary some flowers": *John bought some chocolates and Mary some flowers.*

**Conjunction** is more difficult to define than the previous three relations. It holds between elements of a text when they are ordered temporally, one causes the other, when they describe a contrast or when one elaborates on the other. Examples from [41] will demonstrate these relations. Each of the sentences (a) through (d) should be read immediately following the first sentence in the following example.

*For the whole day he climbed up the mountainside, almost without stopping.*

(a) *Then, as dusk fell, he sat down to rest.* (Temporal order)

(b) *So by night time the valley was far below him.* (Causation)

(c) *Yet he was hardly aware of being tired.* (Contrast)

(d) *And in all this time he met no one.* (Elaboration)

**Lexical Cohesion** holds between two tokens in a text which are either of the same type or are semantically related in a particular way. There are five semantic relations that constitute lexical cohesion.

1. *Reiteration with identity of reference* occurs when a particular entity previously referred to in a discourse is referred to again.

   (a) *John saw a dog.*

   (b) *The dog was a retriever.*

   In above example, (a) refers to a particular dog and (b) refers to the same dog again.

2. *Reiteration without identity of reference* occurs when reference is made to the entire class to which an entity previously referred to in a discourse belongs.

   (a) *John saw a small retriever.*

   (b) *Retrievers are usually large.*

   In above example, (a) refers to one particular member of the set of dogs identified as retrievers while (b) refers to the entire class of retrievers.

3. *Reiteration by means of superordinate* occurs when reference is made to a superclass of the class to which a previously mentioned entity belongs.

   (a) *John saw the retriever.*

   (b) *Dogs are his favorite animals.*

   In above example, (a) refers to a retriever, which is a type of dog, while (b) refers to dogs in general.

4. A *systematic semantic relation* holds when a word, or group of words, has a clearly definable relationship with a previously used word or phrase. For example, both could refer to members of the same set.

   (a) *John likes retrievers.*

   (b) *He doesn't like collies.*

   In above example, (a) refers to retrievers and (b) mentions collies, both of which are subsets of the species of dogs. In this case the relationship can be classified as membership in a particular class.

5. A *nonsystematic semantic relation* holds between two words or phrases in a discourse when they pertain to a particular theme or topic, but the nature of their relationship is difficult to specify. Recognizing this category in a computational system would be more difficult than recognizing the other categories.

   (a) *John spent the afternoon studying in his dormitory room.*

   (b) *He loves attending college.*

   A semantic connection exists between the word "dormitory" in (a) and "college" in (b), but it is hard to classify and unlikely that all such relations, or even the preponderance of them, could be found in a knowledge source in the way that many synonymy relations can be identified using a thesaurus.

Halliday and Hasan's categories overlap to some degree. For example, it can be difficult to distinguish instances of substitution from endophoric reference. Substitution is subtly different in that it relates words of the text, is not a semantic relation and requires the substituted phrase to have the same role as the phrase it substitutes for. This is not the case with reference. Nonetheless, Halliday and Hasan acknowledge that there are instances where more than one category applies equally well.

Halliday and Hasan explain that texts frequently exhibit varying degrees of cohesion in different sections. Obviously, the start of a text cannot be cohesive with preceding sections, nor can the end exhibit cohesion with later sections. In the middle of a text, however, the quantity of cohesion can vary greatly. Some authors, Halliday and Hasan suggest, prefer to alternate between high and low degrees of cohesion.

Texture—which is more frequently called *coherence*—and cohesion are often confused, but differ significantly. Cohesion relates elements of a text and can generally be identified out of context. Texture, however, is a property that applies to an entire text. It is more difficult to define, but can be recognized upon reading a text in its entirety.

# Appendix B

# Tools and Datasets

This appendix briefly describes the datasets and tools that have been used for conducting experiments in this study.

## B.1 Tools

This section briefly describes tools which we have developed for conducting experiments. They are published as open source softwares under the GPL license.

### B.1.1 Wikipedia Processing Toolkit

In the scope of this study, we developed a toolkit for processing the Wikipedia archives. This tool can parse an article in the Wikipedia's special format to get the content in plain text along with metadata of the article, such as category information, table-of-contents, infobox, and so on.

### B.1.2 JSeg – A Java-based Text Segmentation Tool

**JSeg** is a text segmentation tool, which implements our proposed method in text segmentation in Chapter 4. JSeg gets a plain text as the input and produce the segmented text as the output. The text preprocessing steps such as tokenization, sentence boundary detection are done automatically.

**JSeg** has been submitted to CICLing 2011 [75] to be open source as the proof of concepts of our paper[1].

### B.1.3 JCombiner – A Java-based Segment Combination Tool

**JCombiner** is a segment combination tool. It gets a set of segmented texts and produces a flattened tree of segments as the output. It is an implementation of our proposed

[1]JSeg @ CICLing 2011: `http://www.cicling.org/2011/software/174/`

algorithms in Chapter 6.

### B.1.4 HSTGen - HST Generation

**HSTGen** is a tool that implement the HST learning model with supportive knowledge, which has been presented in Chapter 5. It is an extension of **TocGen**[2] which is an implementation of Branavan et al. [17]. HSTGen implements the semi-supervised approach which uses supportive knowledge to provide semantic and topic information to the HST generation learning model.

## B.2 Datasets

### B.2.1 CHOI Dataset

The CHOI dataset contains 700 synthesized documents, in which each document is a concatenation of 10 text segments. Each segment, in turn, is the first $n$ sentences of a randomly selected document from the Brown corpus[3]. Each document is characterized by the range of $n$. The corpus was re-generated by following the description in [22].

Table B.1 shows the number of documents in each set which is characterized by the range of $n$.

Table B.1: CHOI dataset

| Range of $n$ | 3–11 | 3–5 | 6–8 | 9–11 |
|---|---|---|---|---|
| Number of documents | 400 | 100 | 100 | 100 |

### B.2.2 ALG Dataset

This dataset is the content of the textbook "*Introduction to Algorithms*" [27]. This book contains 564 sections in 39 chapters and 7 parts. The depth of the table-of-contents is 4. The authors [17] of this dataset treated the fragment of table-of-contents of each chapter as a small table-of-contents with depth of 3. Therefore, we have 39 small table-of-contents for training and testing.

The table-of-contents of this book can be accessed freely at the website of MIT Press[4].

[2]`http://groups.csail.mit.edu/rbg/code/toc/`

[3]Brown corpus can be freely accessed via NLTK: `http://www.nltk.org`

[4]`http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=8570&mode=toc`

### B.2.3 News Article Dataset

Almost data in these datasets is collected via a news aggregator Techmeme[5]. Articles can easily retrieved by accessing the snapshot of Techmeme. Due to the copyright rules, only the content of the E-Ink dataset is included at the end of this appendix.

#### E-Ink

The E-Ink dataset is a set of news articles written about the presentation of the first color E-Ink e-book reader at the FPD International 2010[6] trade show in Tokyo, November 2010[7]. Table B.2 shows the information of articles with publishers and authors.

Table B.2: E-INK dataset

| **Publisher** | **Title** | **Author** |
|---|---|---|
| The NY Times | Color Comes to E Ink Screens | Eric A. Taub |
| PCWorld | Color E Ink Coming Soon – But When will it Arrive in US? | Brennon Slattery |
| TechSpot | First color E Ink screen coming in 2011 | Emil Protalinski |
| Yahoo! News | Upcoming color E Ink display is ‘milestone,’ but still can’t do video | Ben Patterson |
| ZDNet | Hanvon color e-ink reader to debut at FPD International 2010 in Tokyo | Rachel King |

To provide materials for the future reference, we include the content of above articles at the end of this appendix (page 94). These articles had been segmented manually. Indexing labels of segments are the same with segment’s labels used in experiments in Chapter 6.

#### GReader

This dataset contains articles written about the release of Google Reader for Android platform[8].

[5]http://www.techmeme.com/
[6]http://expo.nikkeibp.co.jp/fpd/2010/english/
[7]http://www.techmeme.com/101108/h1200
[8]http://www.techmeme.com/101201/h1800

Table B.3: GReader dataset

| Publisher | Title | Author |
|---|---|---|
| Google Reader Blog | The Android Google Reader app is here! | Peter Baldwin |
| ReadWriteWeb | Google Reader Gets An Official Android App | Audrey Watters |
| WebProNews | Google Finally Releases Google Reader Android App | Chris Crum |
| AndroidGuys | Official Google Reader App Quietly Hits Android | Scott Webster |
| lifehacker | Official Google Reader App for Android Now Available | Whitson Gordon |

### Zuckerberg

This dataset contains articles written about the event in which Mark Zuckerberg, Founder of Facebook[9], is chosen as the person of the year 2010 by Time[10].

Table B.4: Zuckerberg dataset

| Publisher | Title | Author |
|---|---|---|
| Time | Person of the Year 2010: Mark Zuckerberg | Lev Grossman |
| Fortune | Why Mark Zuckerberg, not Steve Jobs, is Time's Person of the Year | Philip Elmer-DeWitt |
| TechCrunch | Mark Zuckerberg Named Time Magazine's 2010 Person Of The Year | Leena Rao |
| The Guardian | Facebook's Mark Zuckerberg named Time magazine's person of the year | Josh Halliday and Matthew Weaver |
| engadget | Mark Zuckerberg named Time Person of the Year, Jesse Eisenberg sadly not listed | Tim Stevens |

### AppStore

This dataset contains articles written about the release of AppStore for Mac OS X[11].

[9]https://www.facebook.com/
[10]http://www.techmeme.com/101215/h1200
[11]http://www.techmeme.com/101216/h1200

Table B.5: AppStore dataset

| Publisher | Title | Author |
|---|---|---|
| Apple | Apples Mac App Store to Open on January 6 | Apple |
| Fortune | Mac App Store to open on CES Day 1 | Philip Elmer-DeWitt |
| CNet | Apple's Mac App Store to launch January 6 | Don Reisinger |
| PCWorld | Apple Announces Mac App Store Will Open on January 6 | Dan Moren |
| TechCrunch | Apple: Mac App Store Will Be Open For Business In 90 Countries On January 6 | Robin Wauters |

**WordLens**

This dataset contains articles written about the release of an iPhone application, Word Lens, which can translate words inside an image[12].

Table B.6: WordLens dataset

| Publisher | Title | Author |
|---|---|---|
| TechCrunch | Word Lens Translates Words Inside of Images. Yes Really. | Alexia Tsotsis |
| PCWorld | Word Lens Translates Text With Your iPhone Camera | David Chartier |
| MacRumors | Word Lens Offers Real Time Language Translation | Arnold Kim |
| Gizmodo | App of the Day: Word Lens for iPhone | Kyle VanHemert |
| Tech Spot | Word Lens: translate in real time with just your phone's camera | Emil Protalinski |

### B.2.4 WIKI Dataset

WIKI dataset consists of all Wikipedia articles in English [97], which had been created and updated until September 20, 2009. This dataset has been processed by our Wikipedia Processing Toolkit. One can use our toolkit on Wikipedia database[13] to reproduce the dataset.

[12]`http://www.techmeme.com/101217/h1800`
[13]`http://en.wikipedia.org/wiki/Wikipedia:Database_download`

This dataset contains 3,071,253 articles in 817,858 categories. After the pre-processing step, it remains 1,262,389,576 words in 123,340,689 sentences with 6,332,406 distinct words. In practice, to remove noise and reduce the effect of rare words, we use a concise vocabulary with 233,851 distinct words.

# Articles in E-INK Dataset

## Color Comes to E Ink Screens

By Eric A. Taub, The New York Times, November 7, 2010.

**Segment 1-1**

E-book readers are lightweight and use little power, but most have a distinct disadvantage to colorful tablet computers: their black-and-white displays.

But on Tuesday at the FPD International 2010 trade show in Tokyo, a Chinese company will announce that it will be the first to sell a color display using technology from E Ink, whose black-and-white displays are used in 90 percent of the world's e-readers, including the Amazon Kindle, Sony Readers and the Nook from Barnes & Noble.

While Barnes & Noble recently announced a color Nook and the Apple iPad has a color screen, both devices use LCD, the technology found in televisions and monitors. The first color e-reader, from Hanvon Technology, based in Beijing, has an E Ink display.

"Color is the next logical step for E Ink," said Vinita Jakhanwal, an analyst at iSuppli. "Every display you see, whether it's a TV or a cellphone, is in color."

Jennifer K. Colegrove, director of display technologies at DisplaySearch, said it was a milestone moment. "This is a very important development," Ms. Colegrove said. "It will bring e-readers to a higher level."

**Segment 1-2**

E Ink screens have two advantages over LCD – they use far less battery power and they are readable in the glare of direct sunlight.

However, the new color E Ink display, while an important technological breakthrough, is not as sharp and colorful as LCD. Unlike an LCD screen, the colors are muted, as if one were looking at a faded color photograph. In addition, E Ink cannot handle full-motion video. At best, it can show simple animations.

**Segment 1-3**

These are reasons Amazon, Sony and the other major e-reader makers are not yet embracing it. Amazon says it will offer color E Ink when it is ready; the company sees color as useful in cookbooks and children's books, and it offers these books in color through its Kindle application for LCD devices. Sony is also taking a wait-and-see approach.

"On a list of things that people want in e-readers, color always comes up," said Steve Haber, president of Sony's digital reading business division. "There's no question that color is extremely logical. But it has to be vibrant color. We're not willing to give up the true black-and-white reading experience."

But Sriram K. Peruvemba, an E Ink vice president, is not upset by the reluctance of the market leaders to adopt his color technology. "I'm convinced that a lot of times it takes one company to prove the market," Mr. Peruvemba said.

**Segment 1-4**

While barely known in this country, Hanvon is the largest seller of e-readers in China. Its founder and chairman, Liu Yingjian, says Hanvon has a 78 percent share of the Chinese market.

Hanvon's first product using a 9.68-inch color touch screen will be available this March in China, starting at about \$440. The price is less than an iPad in China, which sells for about \$590. It will be positioned as a business product, with Wi-Fi and 3G wireless connectivity.

"It's possible that we'll sell this in the U.S. as well," Mr. Liu said. Hanvon sells other products, like tablets and e-readers, to Americans online and through Fry's, a regional electronics chain.

**Segment 1-5**

E Ink, based in Cambridge, Mass., was bought by Prime View Holdings of Taiwan in 2009 and was recently renamed E Ink Holdings. To create the color image, E Ink uses its standard black-and-white display overlaid with a color filter. As a result, battery life is the same as its black-and-white cousins, measured in weeks rather than hours, as with the iPad. The color model from Hanvon can be easily read in bright light, although the color filter does reduce the brightness.

**Segment 1-6**

The Hanvon e-reader is not intended to be a multifunction competitor to the iPad, but rather a dedicated reading device, like the Kindle. Ms. Colegrove of DisplaySearch said these types of lower-cost products should continue to gain market share, growing from four million units sold worldwide in 2009 to 14 million units by 2011. At the same time, slate-type devices like the iPad will increase from one million in 2009 to 40 million in 2011, she predicts.

"Color is absolutely critical for E Ink," said James McQuivey, an analyst at Forrester Research. "Without it, they'll either be replaced by LCD displays or other competitors."

## Color E Ink Coming Soon – But When will it Arrive in US?

By Brennon Slattery, PCWorld, November 9, 2010.

**Segment 2-1**

Hanvon Technology, which is based in Beijing, is expected to unveil the first color E Ink reading device at the FPD International 2010 trade show in Tokyo tomorrow, according to the New York Times.

**Segment 2-2**

The thus-far unnamed device features a 9.6-inch color E Ink display, Wi-Fi, and 3G connectivity; and will hit the market in March 2011 for the equivalent of US$440 – roughly $150 less than the iPad's price in China.

Hanvon's founder and chairman, Liu Yingjian, told the New York Times that "It's possible that we'll sell this in the U.S. as well."

**Segment 2-3**

Up until now, E Ink technology has been solely in high-contrast black and white. Most companies have adhered to this colorless technology, but with the iPad's growing popularity as an e-reader – and Barnes and Noble's – it looks as though others will need to adopt color E Ink or risk becoming obsolete. (I'm looking at you, Amazon.)

Amazon has been clinging to E Ink since inception, and Amazon CEO Jeff Bezos said that while the company has plans to create one eventually, the color Kindle is "still a long way out."

Bezos also said that he's seen "several [color touchscreens] in the laboratory, but they are not quite ready for production." Nothing yet has matched the readability of E Ink tech.

Sony, with its oft-forgotten e-reader, told the New York Times that it doesn't have concrete plans to delve into color either. "On a list of things that people want in e-readers, color always comes up. There's no question that color is extremely logical. But it has to be vibrant color. We're not willing to give up the true black-and-white reading experience," said Steve Haber, president of Sony's digital reading business division.

The color version of Barnes and Noble's Nook e-reader has a 7-inch backlit touch screen with 16 million colors – but the new Nook is decidedly not a traditional e-reader. Rather, it's a tablet marketed as an e-reader. While Barnes and Noble plans on having a robust app store, this is directly positioning its Nook beside the iPad – and that's not exactly wise.

I'm hoping that Hanvon's announcement has stirred conversation at Amazon's headquarters. Color e-readers are here to stay, and now that color E Ink technology is out in the wild, Amazon ought to be the first U.S. company to make it happen, and perhaps put LCD backlit tablets to shame.

## First color E Ink screen coming in 2011

By Emil Protalinski, TechSpot, November 8, 2010

**Segment 3-1**

Color e-readers have existed in prototype form for a while now but soon they'll be hitting the market en masse. It will all begin with Chinese e-reader maker Hanvon, a company that plans to ship the first color reader next year. Hanvon's device sports a 9.68-inch color touch screen, Wi-Fi, and 3G. It will retail in China in March 2011 for about $440. "It's possible that we'll sell this in the US as well," Liu Yingjian, Hanvon's chairman told The New York Times. Even if Hanvon doesn't do it, one of their competitors definitely will.

**Segment 3-2**

The e-reader uses a standard E Ink screen with a color filter. As a result, it still has the same low-power, lightweight, high-readability characteristics of its black-and-white brethren. The downside is that its screen is pretty static: color images and illustrations are okay (basic animation might be possible), but full-motion video is definitely out of the question. Furthermore, a lack of backlight means the colors won't be as bright as an LCD screen. Other features of the device have yet to be revealed; Hanvon is known for its handwriting technology, but it doesn't include it in all of its e-readers.

**Segment 3-3**

Color isn't as important in reading as it is in media entertainment and gaming. Will color illustrations be enough, or will readers instead choose the more powerful tablets with LCD screens? Chances are that consumers will want everything: e-books with color, media entertainment, video games, all with zero glare and the low power consumption that translates to longer battery life. Oh and a lower price tag wouldn't hurt. Right now that's not possible, so what tradeoffs will you settle for?

## Upcoming color E Ink display is 'milestone,' but still can't do video

By Ben Patterson, Yahoo! News, November 8, 2010

**Segment 4-1**

A Chinese company is primed to launch a color e-reader early next yearand unlike the recent Nook Color from Barnes & Noble, the new device will have an actual E Ink display (similar to those on the Amazon Kindle and the Sony Reader) rather than going the LCD way (like the iPad).

But the upcoming Hanvon e-reader, slated to be unveiled Tuesday at a Tokyo trade show (according to the New York Times), will also come saddled with several of the inherent drawbacks of current E Ink technologyparticularly a glacial refresh rate that renders smooth, full-motion video next to impossible.

**Segment 4-2**

Hanvon's 9.68-inch, touch-enabled e-reader is poised to go on sale next March in China for about $440almost the same price as the 16GB iPadaccording to the Times.

**Segment 4-3**

The slate uses a color display developed by E Ink, which manufacturers the black-and-white e-paper display on such current e-readers as the Kindle, the Sony Reader and Barnes & Noble's original, monochrome Nook.

The secret, the Times reports, is a color filter that sits atop the usual black-and-white E Ink display.

The Hanvon reader will also support Wi-Fi and 3G, says the Times, and it'll be primarily aimed at business users.

Of course, it's not like we haven't already seen color e-readers here in the U.S. There's the iPad, of course, not to mention Barnes & Noble's new Android-based Nook Color.

But the iPad and Nook Color tablets use traditional LCD displays, which can be hard to read outdoors and are battery hogs compared with E Ink readers like the Kindle, which keep going and going... and going, for days and even weeks at a time.

The Hanvon color E Ink slate will also have extra-long battery life, the Times reports, and it will be nearly as easy to read outdoors as current black-and-white E-Ink devices.

Just don't expect to watch episodes of "Mad Men" on the Hanvon. As with the Kindle, the Sony Reader and the first Nook, the E Ink display on the Hanvon reader can't refresh nearly as fast as an LCD screen can, resulting in "simple animations" at best and no video, of course, says the Times.

**Segment 4-4**

Even the color images themselves on the Hanvon are "muted" like a "faded color photograph," and the color filter "does reduce the brightness" on the E Ink display, the story continues.

So while a commercially available color E Ink reader probably is a "milestone" in the e-reader market, as one analyst told the Times, it'll still represent a trade-offone that some players in the e-reader field, like Amazon, still appear unwilling to make.

Back in July, when Amazon unveiled its revamped Kindle, I asked Amazon reps when a color and/or touchscreen Kindle might be on the wayand the answer I got was that for the "vast majority" of readers, a sharp black-and-white screen is "a feature, not a bug." The Amazon spokesperson also argued that an extra layer of touch-sensitive glass would cut down on the contrast of the screen, which would be too high a price to pay given that Kindle users spend most of their time simply tapping the "next page" button over and over.

So for now, it appears we're still years away from the holy grail of display technology: a screen that looks great outside, works for days and weeks on a single charge, and is fast enough to display razor-sharp video, just like LCD. At least the Hanvon e-reader sounds like a step in the right direction, albeit a small one.

## Hanvon color e-ink reader to debut at FPD International 2010 in Tokyo

By Rachel King, ZDNet, November 8, 2010

**Segment 5-1**

The Nook Color is on its way, but maybe there will be some more competition in the color e-book reader market soon.

Hanvon Technology is all set to debut a 10-inch e-reader with a colorized electronic ink display at the FPD International 2010 trade show in Tokyo on Tuesday.

**Segment 5-2**

According to The New York Times, the Hanvon e-reader is "not intended to be a multifunction competitor to the iPad, but rather a dedicated reading device, like the Kindle." The article cites a lot of the pros and cons when it comes to e-ink versus and LCD. The biggest hindrance with color LCD panels is glare, making the devices less versatile when it comes to location.

**Segment 5-3**

But I have to agree that color is just where this market is headed. It will be costly at first (as evident by the initial $249 price tag attached to the Nook Color), but eventually that will drop. So if we can get color e-ink technology rolling faster, I can't wait to see what might come out in the next year.

Sporting both Wi-Fi and 3G support, the color e-ink device is expected to first launch in Hanvon's native China next year for $440. According to the company's founder and chairman, Liu Yingjian, it is "possible" that we'll see this one sold in the United States. But it's going to need a serious price slashing for any chance of success.

# Bibliography

[1] S. Abney. Understanding the yarowsky algorithm. *Computational Linguistics*, 30(3):365–395, 2004.

[2] S. Abney. *Semisupervised Learning for Computational Linguistics.* Chapman and Hall/CRC, 2007.

[3] E. Alpaydin. *Introduction to Machine Learning.* The MIT Press, second edition, 2010.

[4] R. Angheluta, R. D. Busser, and M.-F. Moens. The use of topic segmentation for automatic summarization. In *Proceedings of the Workshop on Text Summarization in Conjunction with the Annual Meeting of the Association of Computational Linguistics (DUC)*, pages 11–12, Philadelphia, Pennsylvania, USA, 2002.

[5] M. Banko, V. O. Mittal, and M. J. Witbrock. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 318–325, Hong Kong, 2000.

[6] R. Barzilay and M. Elhadad. Using lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL*, pages 10–17, Madrid, Spain, 1997.

[7] S. Basu. *Semi-supervised Clustering: Probabilistic Models, Algorithms and Experiments.* PhD thesis, The University of Texas at Austin, August 2005.

[8] D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210, 1999.

[9] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):1–30, 2010.

[10] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine learning*, pages 113–120, Pittsburgh, Pennsylvania, USA, 2006.

[11] D. M. Blei and J. D. Lafferty. A correlated topic models of science. In *The Annals of Applied Statistics*, volume 1, pages 17–35, 2007.

[12] D. M. Blei and P. J. Moreno. Topic segmentation with an aspect hidden markov model. In *Proceedings of the 24th Annual International ACM SIGIR Conference*

*on Research and Development in Information Retrieval*, pages 343–348, New York, NY, USA, 2001. ACM.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Machine Learning Research*, 3:993–1022, 2003.

[14] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 19–26, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[15] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

[16] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[17] S. R. K. Branavan, P. Deshpande, and R. Barzilay. Generating a table-of-contents. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 544–551, Prague, Czech Republic, 2007.

[18] P. F. Brown, P. V. Desouza, R. L. Mercer, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[19] L. Carroll. Evaluating hierarchical discourse segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 993–1001, Los Angeles, California, June 2010. Association for Computational Linguistics.

[20] E. Charniak. *Statistical Language Learning*. The MIT Press, 1996.

[21] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems*, pages 431–441, 1993.

[22] F. Y. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of the First Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 26–33, Seattle, USA, 2000.

[23] F. Y. Y. Choi, P. Wiemer-Hastings, and J. Moore. Latent semantic analysis for text segmentation. In L. Lee and D. Harman, editors, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 109–117, 2001.

[24] K. W. Church. Char_align: A program for aligning parallel texts at the character level. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics (ACL)*, pages 1–8, Morristown, NJ, USA, 1993.

[25] M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 263–270, Philadelphia, USA, July 2002.

[26] M. Collins and B. Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL)*, pages 111–118, Barcelona, Spain, 2004.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, second edition, 2001.

[28] H. Daumé III and D. Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 169–176, Bonn, Germany, 2005.

[29] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *The American Society for Information Science*, 41:391–407, 1990.

[30] A. Dielmann and S. Renals. Multistream dynamic bayesian network for meeting segmentation. In S. Bengio and H. Bourlard, editors, *Machine Learning for Multimodal Interaction*, volume 3361 of *Lecture Notes in Computer Science*, pages 76–86. Springer Berlin / Heidelberg, 2005.

[31] B. Dorr, D. Zajic, and R. Schwartz. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop*, pages 1–8, Edmonton, Canada, 2003.

[32] S. Dubnov, R. El-Yaniv, Y. Gdalyahu, E. Schneidman, N. Tishby, and G. Yona. A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles. *Machine Learning*, 47(1):35–61, 2002.

[33] J. Eisenstein. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 353–361, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[34] J. Eisenstein and R. Barzilay. Bayesian unsupervised topic segmentation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 334–343, Honolulu, Hawaii, 2008.

[35] O. Ferret. Finding document topics for improving topic segmentation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 480–487, Prague, Czech Republic, June 2007.

[36] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[37] M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, pages 562–569, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[38] S. A. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 327–334, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[39] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004.

[40] A. Haghighi and L. Vanderwende. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[41] M. A. K. Halliday and R. Hasan. *Cohesion in English.* Longman Pub Group, 1976.

[42] M. A. Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 9–16, Las Cruces, New Mexico, USA, 1994.

[43] M. A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, 1997.

[44] G. Heinrich. Parameter estimation for text analysis. Technical report, University of Leipzig, Germany, 2005.

[45] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

[46] X. Ji and H. Zha. Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 322–329, Toronto, Canada, 2003.

[47] R. Jin and A. G. Hauptmann. A new probabilistic model for title generation. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, pages 1–7, Taipei, Taiwan, 2002.

[48] T. Joachims. Transductive inference for text classi
cation using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pages 200–209, 1999.

[49] K. S. Jones. Automatic summarising: The state of the art. *Information Processing and Management*, 43(6):1449–1481, 2007.

[50] D. Jurafsky and J. H. Martin. *Speech and Language Processing.* Prentice Hall, second edition, 2008.

[51] S. D. Kamvar, D. Klein, and C. D. Manning. Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 283–290. Morgan Kaufmann Publishers Inc., 2002.

[52] D. Kauchak and F. Chen. Feature-based segmentation of narrative documents. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, pages 32–39, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[53] T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics (ACL-HLT)*, pages 595–603, Columbus, Ohio, USA, 2008.

[54] P. Liang and M. Collins. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, 2005.

[55] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS)*, pages 25–26, Barcelona, Spain, 2004.

[56] D. Lin and X. Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1030–1038, Suntec, Singapore, August 2009.

[57] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[58] I. Malioutov. Minimum cut model for spoken lecture segmentation. Master's thesis, Massachusetts Institute of Technology, 2006.

[59] I. Malioutov and R. Barzilay. Minimum cut model for spoken lecture segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 25–32, Sydney, Australia, 2006.

[60] I. Mani and M. T. Maybury. *Advances in Automatic Text Summarization.* The MIT Press, 1999.

[61] C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing.* The MIT Press, 1999.

[62] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

[63] S. Martin, J. Liermann, and H. Ney. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19–37, 1998.

[64] K. R. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, J. L. Klavans, A. Nenkova, C. Sable, B. Schiffman, and S. Sigelman. Tracking and summarizing news on a daily basis with columbia's newsblaster. In *Proceedings of the 2nd International Conference on Human Language Technology Research*, pages 280–285, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[65] B. Merialdo. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–171, 1994.

[66] S. Miller, J. Guinness, and A. Zamanian. Name tagging with word clusters and discriminative training. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 337–342, Boston, Massachusetts, USA, 2004.

[67] H. Misra, F. Yvon, J. M. Jose, and O. Cappe. Text segmentation via topic modeling: an analytical study. In *CIKM '09: Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 1553–1556, 2009.

[68] J. Morris and G. Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48, 1991.

[69] P. V. Mulbregt, I. Carp, L. Gillick, S. Lowe, and J. Yamron. Text segmentation and topic tracking on broadcast news via a hidden markov model approach. In *Proceedings of the 5th International Conference on Spoken Language Processsing*, pages 2519–2522, Sydney, Australia, 1998.

[70] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin. Automatic evaluation of topic coherence. In *Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108, Los Angeles, California, USA, June 2010.

[71] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14:849–856, 2002.

[72] L. M. Nguyen, X. B. Ngo, V. C. Nguyen, M. P. Q. Nhat, and A. Shimazu. A semi-supervised learning method for vietnamese part-of-speech tagging. In *Proceedings of the Second International Conference on Knowledge and Systems Engineering (KSE)*, pages 141–146, Ha Noi, Viet Nam, October 2010.

[73] V. C. Nguyen, L. M. Nguyen, and A. Shimazu. A semi-supervised approach for generating a table-of-contents. In *Proceedings of the 7th International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 313–318, Borovets, Bulgaria, 2009.

[74] V. C. Nguyen, L. M. Nguyen, and A. Shimazu. A semi-supervised model for table-of-contents generation. In *Proceedings of the 11th Conference of the Pacific Association for Computational Linguistics (PACLING)*, pages 147–152, Sapporo, Japan, 2009.

[75] V. C. Nguyen, L. M. Nguyen, and A. Shimazu. Improving text segmentation with non-systematic semantic relation. In *Proceedings of the 12th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, volume Lecture Notes in Computer Science (LNCS) 6608–6609, Tokyo, Japan, February 2011. Springer-Verlag Berlin Heidelberg.

[76] V. C. Nguyen, L. M. Nguyen, and A. Shimazu. Learning to generate a table-of-contents with supportive knowledge. *IEICE Transactions on Information and Systems (Special Section on Knowledge Discovery, Data Mining and Creativity Support System)*, 94-D(3):1–9, March 2011.

[77] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 9th International Conference on Information and Knowledge Management*, pages 86–93, 2000.

[78] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3):103–134, 2000.

[79] J. Otterbacher, D. Radev, and O. Kareem. Hierarchical summarization for delivering information to mobile devices. *Information Processing and Management*, 44(2):931–947, 2008.

[80] R. J. Passonneau and D. J. Litman. Discourse segmentation by human and automated means. *Computational Linguistics*, 23(1):103–139, 1997.

[81] F. Pereira and Y. Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 128–135, 1992.

[82] L. Pevzner and M. A. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36, 2002.

[83] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceeding of the 17th International Conference on World Wide Web*, pages 91–100, 2008.

[84] J. M. Ponte and W. B. Croft. Text segmentation by topic. In *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, pages 113–125, London, UK, 1997. Springer-Verlag.

[85] D. Radev, J. Otterbacher, A. Winkel, and S. Blair-Goldensohn. Newsinessence: Summarizing online news topics. *Communications of the ACM*, 48:95–98, October 2005.

[86] J. C. Reynar. An automatic method of finding topic boundaries. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, pages 331–333, Las Cruces, New Mexico, USA, 1994.

[87] J. C. Reynar. *Topic Segmentation: Algorithms and Applications.* PhD thesis, University of Pennsylvania, 1998.

[88] S. W. Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 42(1):97–101, 2000.

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

[90] E. Shriberg, A. Stolcke, D. Hakkani-Tür, and G. Tür. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, 32(1-2):127–154, 2000.

[91] M. Slaney and D. Ponceleon. Hierarchical segmentation: Finding changes in a text signal. In *Proceedings of the SIAM Text Mining 2001 Workshop*, pages 6–13, Chicago, IL, 2001.

[92] D. Sontag and D. M. Roy. Complexity of inference in topic models. In *Proceedings of the Wordshop on Topic Models: Text and Beyond in Neural Information Processing Systems Conference (NIPS)*, Vancouver, B.C., Canada, 2009.

[93] C. Sporleder and M. Lapata. Broad coverage paragraph segmentation across languages and domains. *ACM Transactions on Speech and Language Processing (TSLP)*, 3(2):1–35, 2006.

[94] J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 384–394, Uppsala, Sweden, 2010.

[95] A. Ushioda. Hierarchical clustering of words and application to nlp tasks. In *Proceedings of the 4th Workshop on Very Large Corpora*, pages 28–41, 1996.

[96] M. Utiyama and H. Isahara. A statistical model for domain-independent text segmentation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 499–506, Toulouse, France, 2001.

[97] J. Wales and L. Sanger. Wikipedia – the free encyclopedia. `http://en.wikipedia.org`, 2001.

[98] H. M. Wallach, D. Mimno, and A. McCallum. Rethinking lda: Why priors matter. In *Proceedings of the Wordshop on Topic Models: Text and Beyond in Neural Information Processing Systems Conference (NIPS)*, pages 1973–1981, Vancouver, B.C., Canada, 2009.

[99] R. Wang, J. Dunnion, and J. Carthy. Machine learning approach to augmenting news headline generation. In *Proceedings of 2nd International Joint Conference on Natural Language Processing (IJCNLP)*, pages 155–160, Korea, 2005.

[100] X. Wang, A. McCallum, and X. Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of the 7th IEEE International Conference on Data Mining*, pages 697–702, Omaha, Nebraska, USA, 2007.

[101] M. J. Witbrock and V. O. Mittal. Ultra-summarization: A statistical approach to generating highly condensed non-extractive summaries. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 315–316, Berkeley, California, United States, 1999.

[102] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the 4th ACM Conference on Digital Libraries*, pages 254–255, Berkeley, California, United States, 1999.

[103] Y. Yaari. Segmentation of expository texts by hierarchical agglomerative clustering. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 59–65, Bulgaria, 1997.

[104] Y. Yaari. Texplore - exploring expository texts via hierarchical representation. In *Proceedings of the Workshop on Content Visualization and Intermedia Representations (CVIR) in COLING-ACL '98*, pages 25–32, Montreal, Quebec, Canada, 1998.

[105] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, pages 189–196, 1995.

[106] D. M. Zajic, B. Dorr, and R. Schwartz. Bbn/umd at duc-2004: Topiary. In *Proceedings of North American Chapter of the Association for Computational Linguistics Workshop on Document Understanding (DUC)*, pages 112–119, Boston, MA, USA, 2004.

# Publications

## Journals

[1] V. C. Nguyen, L.M. Nguyen, and A. Shimazu. Learning to Generate a Table-of-Contents with Supportive Knowledge. *IEICE Transactions on Information and Systems (Special Section on Knowledge Discovery, Data Mining and Creativity Support System)*, Volume 94-D, Number 3, pages 1–9, March 2011.

## Refereed International Conferences

[2] V. C. Nguyen, L. M. Nguyen, and A. Shimazu. Improving Text Segmentation with Non-systematic Semantic Relation. *Computational Linguistics and Intelligent Text Processing (CICLing 2011)*, Lecture Notes in Computer Science (LNCS), Volume 6608–6609, Springer-Verlag Berlin Heidelberg, February 2011. To appear.

[3] L. M. Nguyen, X. B. Ngo, V. C. Nguyen, Q. N. M. Pham, and A. Shimazu. A Semi-supervised Learning Method for Vietnamese Part-of-Speech Tagging. In *Proceedings of the Second International Conference on Knowledge and Systems Engineering (KSE 2010)*, pages 141–146, Ha Noi, Viet Nam, October 2010.

[4] V. C. Nguyen, L. M. Nguyen, and A. Shimazu. A Semi-supervised Model for Table-of-Contents Generation. In *Proceedings of the 11th Conference of the Pacific Association for Computational Linguistics (PACLING 2009)*, pages 147–152, Sapporo, Japan, September 2009.

[5] V. C. Nguyen, L. M. Nguyen, and A. Shimazu. A Semi-supervised Approach for Generating a Table-of-Contents. In *Proceedings of the 7th International Conference on Recent Advances in Natural Language Processing (RANLP 2009)*, pages 313–318, Borovets, Bulgaria, September 2009.