DOCUMENT SEMANTIC REPRESENTATION: AN ALGEBRAIC TOPOLOGICAL APPROACH

by

Fanchao Meng

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Summer 2019

DOCUMENT SEMANTIC REPRESENTATION: AN ALGEBRAIC TOPOLOGICAL APPROACH

by

Fanchao Meng

Approved: ____
Kathleen McCoy, Ph.D.
Chair of the Department of Computer & Information Sciences

Approved: ____
Levi T. Thompson, Ph.D.
Dean of the College of Engineering

Approved: ____
Douglas J. Doren, Ph.D.
Interim Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ______________________________

Errol Lloyd, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ______________________________

B. David Saunders, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ______________________________

Christopher Rasmussen, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ______________________________

Chad Giusti, Ph.D.
Member of dissertation committee

Acknowledgements

First, I would like express my deepest appreciation to my advisor, Dr. Errol Lloyd. Dr. Lloyd has guided me through numerous difficulties in the past seven years. His wisdom, support and patience have formed a constant source of inspiration for me. This dissertation is a direct result from our myriad discussions. Dr. Lloyd helped me turn my imaginary ideas into real methods and models, and our work finally opens a new branch of document semantic representation. His contribution to this dissertation and his advising to me are appreciated beyond words.

I would like to thank Dr. Kathleen McCoy. She and Dr. Lloyd have provided a strong funding support to me. Without their support, I could never complete my doctoral degree.

I also appreciate DARPA and the funded SocialSim team at the University of Virginia (previously at Virginia Tech) that has been collaborating with me. Their resource and support made our research smoother. Particularly, I would like to extend my sincere gratitude to Dr. Madhav Marathe and Dr. Samarth Swarup. Dr. Marathe offered me a research associate position at the Biocomplexity Institute of Virginia Tech in 2018 summer. This job helped me make significant progress on my research. Dr. Swarup and I have had many fruitful discussions, and our work has led to some very interesting results in applying our document semantics comparison method to social dynamics applications. Dr. Marathe and Dr. Swarup have also offered me a postdoctoral researcher position at the University of Virginia. We will keep exploring cutting-edge interdisciplinary areas between social dynamics and natural language processing. I am also grateful to the colleagues at the University of Virginia. Dr. Mark Orr, Dr. S.S. Ravi, Dr. Abhijin Adiga, Dr. Anil Vullikanti, Dr. Christopher Kuhlman, Dr. Parantapa Bhattacharya, Mandy Wilson and Jim Walke, they are always kind and patient to help me with anything I needed.

I also sincerely appreciate my dissertation committee members, Dr. David Saunders, Dr. Christopher Rasmussen and Dr. Chad Giusti. They have given many valuable suggestions to my research and this dissertation. These suggestions have made this dissertation more rigorous and complete.

I would like to thank all faculty members who ever helped me. Particularly, Dr. Gregory Lynch, he is the one who always stands behind me and has a faith in me. I also thank Dr. John Case, Dr. Paul Amer, Dr. Adarsh Sethi, Dr. Hagit Shatkay, Dr. Ben Carterette, Dr. Sandra Carberry, Dr. Stephan Bohacek, Dr. Felix Lazebnik, Dr. Rakesh and Dr. Qing Xiang. I have learned a lot in your classes. I am also grateful to Dr. Chen-Chung Shen, Dr. Stephen Siegel, Dr. Daniel Chester, Dr. Keith Decker, Dr. Greg Silber, Dr. Debra Yarrington and Dr. Jinwei Cao. I am honored that I worked as a teaching assistant with you. I also thank Dr. Haiping Wu who has shared his house for me to live and provided great help for me.

Thanks should also go to my best friends who always support me, Yifan Li, Tianzhao Yao, Lu Zhang, Zhuo Li, Qingchun Han, Jiayi Zhao, Sheng Zhu, Wanxin Li, Fan Cheng, Yucheng Wang, Gang Wang, Chunbo Song and many other friends. You have made my last seven years enjoyable.

Last, and most important, my deepest gratitude are to my family and my love. There is no word that can express my gratitude to my mother, Qiuping Zhang, and my father, Qingyi Meng. You support me with your entire life. You have a constant faith in me since I was born. Your love is in every meal that you made for me, and your love is in every tear that you do not want me to see. Finally, I feel extremely lucky that I met Yan, my love. She has given me not only a considerable amount of irreplaceable help on my research and dissertation but also an irreplaceable life.

Table of Contents

List of Tables

List of Figures

Abstract

Document semantics representation bridging human-written texts and machine text understanding is one of the most cutting-edge areas in Natural Language Processing, Information Retrieval and Computational Linguistics. Geoffrey Hinton profiles the future of machine text understanding with a picture of human-level machine reasoning rooted in document semantics representations[1].

Over several decades document semantics representation research has formed into two branches, *distributional semantics* and *syntactic structures*. *Distributional semantics* achieves document semantics representations by representing lexical semantics through a following idea: "You shall know a word by the company it keeps"[2]. This is implemented by using statistical information on word (co-)occurrences. To date, *word embedding* techniques have dominated this branch, and are mostly based on *neural network* models. Document semantics representations then are built upon word embedding vectors, and thus are also vectorized, which is convenient for downstream applications. On the other hand, document semantics representations rooted in syntactic structures represent syntactic relations between words in a sentence such as syntactic dependencies and constituent structures. Critical in reflecting document

[1] Readers are referred to a 2015 speech to the Royal Society in London given by Dr Geoffrey Hinton

[2] This is a famous quote from J. R. Firch

semantics such document semantics representations capture more sophisticated relations between words than statistical information of word (co-)occurrences. Document semantics representations in this branch are typically designed as trees or sets of relations of sentences. Unfortunately, both of these branches have critical drawbacks. Most word embedding methods have to confront subtleness in training, while most syntactic-structure-based methods have to fill the gap between sentence semantic representation and document semantic representation.

Considering the pros and cons of these two branches, naturally designing document semantics representations that have vector forms, that utilize *syntactic structures*, and that do not require training is the objective of this dissertation. To address this problem, we introduce a new element, named *generalized phrase* that is extracted from constituency-based parse trees, as the major ingredient to construct our document semantics representations. Toward establishing the effectiveness of generalized phrases in reflecting document semantics, we propose a new *document semantics comparison* method that extracts and utilizes generalized phrases. This method is named *DSCTP*, and is developed from algebraic topology particularly focusing on persistence. We test *DSCTP* on both document semantics comparison tasks and *hard document clustering* tasks. The experimental results show that, on the document semantics comparison tasks, *DSCTP* can provide competitive, and some cases better, performance than state-of-the-art methods; and, on the hard document clustering tasks, *DSCTP* can significantly outperform state-of-the-art methods. Based on generalized phrases extracted by *DSCTP* we propose two new document semantics representations in vector forms that do not require training. One is constructed on a clustering of generalized phrases, and the other is constructed by using *graph signal processing* techniques. We name them *abstract phrase vector* and *generalized phrase graph signal* respectively. The experimental results on the hard document clustering tasks show that both abstract phrase vector and generalized phrase graph signal perform as well as, and some cases outperform, state-of-the-art document semantics representations.

Chapter 1
INTRODUCTION

The core task of this dissertation is providing a new computational representation for a document to reflect the semantics of that document (i.e. **document semantic representation (DSR)**). In this dissertation, **documents** are texts written in English consisting of one or more sentences[1], and the **semantics** of a document is the meaning of the document that is rationally understood by humans.

To fully understand the core task, we briefly explain the term, *semantics*[2], a key concept in cognitive science and linguistics [125] [90] [35]. In this dissertation, the semantics of a text is the literal meaning of that text. Note that the literal meaning is independent of the context of that text. Thus, semantics in this dissertation is explicitly distinguished from *pragmatic meaning* [35] which is over and above the literal meaning and is dependent on context[3]. In addition, typically a semantics has a common-sense understanding for humans. In other words, most of the time, the majority will agree with each other on understanding a semantics[4]. Such common-sense understandings are fundamental in many studies involving human efforts such as [82], [78] and [89]. Semantics in this dissertation is interpreted from the common-sense understanding point of view. In addition, semantics should be distinguished from another term, *topic*.

[1] "Irregular" texts such as program code and table data are not in consideration in this dissertation.

[2] Readers interested in further details about semantics are referred to [125] [90] and [35]

[3] For example, the saying *"go flush myself down the toilet"* in most cases actually means that one feels really embarrassed and would like to run away immediately, which is obviously not its literal meaning. Such pragmatics will not be considered in this dissertation.

[4] Theoretically, *semantics* is not a definite object [115]. For example, for a document, although people can agree on the lexical meaning of each word and every grammar involved in the document, they may not have the same understanding of the meaning of the document.

Loosely speaking, a topic is a more general level of meaning, a category of interest and usually is represented as a set of words. Documents that have the same topic can have significantly different semantics.

The reminder of this chapter is organized as follows: First, we summarize the major contributions in this dissertation (Section 1.1); and second, we explain why *DSR* is an interesting topic and how our work is motivated (Section 1.2).

1.1 Major Contributions

Our first major contribution is a method based on *topological persistence* [42] to compute the semantic similarity between two documents. This methods takes two documents as inputs, and returns a real value reflecting the semantic similarity between the documents. We name it **DSCTP**, where *DSCTP* is short for *document semantics comparison based on topological persistence*. We note that *DSCTP* is a pairwise similarity method that works *without* representing each individual document. We compare the performance of *DSCTP* with a set of state-of-the-art methods for document semantics comparison. These methods are based on state-of-the-art semantic representation methods including *Word2Vec* [98] [80] *Doc2vec* [80], *NASARI* [20], *GloVe* [110], *fastText* [15], *LexVec* [126] and *Sent2Vec* [107]. Among these methods, *Word Mover Distance (WMD)* [77] will be integrated with *Word2Vec* to form a document semantics comparison method, and the others will be integrated with cosine similarity to compare document semantics. Our experimental results show that *DSCTP* can outperform or perform similarly to each of these methods. In addition, we apply *DSCTP* and all of the above methods to *document clustering*, and the experimental results show that the *DSCTP*-based document clustering method outperforms most of the document clustering methods based on the above document comparison methods. *DSCTP* is detailed in Chapter 2.

Our second major contribution is a single document representation and a corresponding method to construct the representation. More specifically, we represent the semantics of each document as a real-valued vector. The corresponding construction method is based on *DSCTP*. We name this representation **abstract phrase vector (APV)**. We evaluate the effectiveness of *APV* by applying it to *document clustering*, and the experimental results show that the *APV*-based document clustering method provides similar, and in some cases better, performance as the state-of-the-art methods. *APV* and the construction method are discussed in Chapter 3.

Our third major contribution is another single document representation and a corresponding construction method based on a combination of *DSCTP* and *Graph Signal Processing (GSP)* [135] [128] [129]. This work is one of the vanguards[5] of applying *GSP* in *Natural Language Processing (NLP)* problems. We name this representation *generalized phrase graph signal (GPGS)*. We also apply it to *document clustering*, and the experimental results show that the *GPGS*-based document clustering method can also perform as well as, and in some cases outperform, state-of-the-art methods. *GPGS* and the construction method are discussed in Chapter 4. The experiments on document clustering are discussed in Chapter 5.

Our final major contribution is that we apply *DSCTP*, *APV* and *GPGS* to a problem in *Social Dynamics* [41]. This problem asks, in a social network, when a user issues a message (e.g. a tweet), then who is likely to propagate this message. Our methods take advantage of *DSCTP*, *APV* and *GPGS* to attack the problem. We detail this problem and our methods in Chapter 6.

1.2 Motivations & Related Work

DSR is an interesting and critical topic in *Information Retrieval*, *NLP* and *Text Mining* [165] as it is a significant ingredient in solving quite a number of problems in these areas such as *document clustering* [145], *document semantics comparison* [51], *document classification* [24] and *document indexing* [47]. To date, state-of-the-art *DSRs* are mostly either relying on *neural-network-based word embedding (NNWE)* models

[5] To date, the only previous work applying *GSP* techniques to *NLP* problems is [152], in which *Hadamard code* [120] is used to represent sentence meanings.

which have a number of general drawbacks, or rooted in traditional methods in computational linguistics, particularly utilizing syntactic structures (i.e. syntactic relations between words in a sentence), which are usually not convenient to use or are specific to particular problems For these reasons, we aim to design alternative *DSRs* which are not based on *neural network* models and are universal to as many problems as possible.

1.2.1 Existing DSRs

In this section, we briefly review the pros and cons of a collection of state-of-the-art *DSRs* including both neural-network-based methods and syntactic-structure-based methods. These reviews will help for understanding why we aim to design alternative *DSRs*. Then we review a particular use of *suffix trees* [158] in document clustering, and explain how this method motivates our work.

1.2.1.1 NNWE-Based Methods

Geoffrey Hinton, one of the pioneers in *deep learning* [81], proposed a concept named "*thought vector*" which represents a thought as a vector encoding the relations between this thought and other thoughts [61]. Representing document semantics as vectors is a special case of this concept, and this is one of the primary goals of recent neural-network-based natural language processing research. However, to date, few works have been approaching this goal. *Doc2Vec* [80] and *Skip-Thoughts* [75] are two typical endeavors. Most state-of-the-art semantic representations (mostly for words) are *neural network* and *vector space model (VSM)* -based methods (e.g. *Word2vec* [98] [97]). Such methods have been applied in various applications obtaining impressive results in problems such as *semantic textual similarity* [79] and *sentiment analysis* [80]. The cornerstone of these methods is *word embedding* [43] which is a category of techniques representing each word as a vector of features and forming a *vector space* over the features. Then each word is "embedded" into the vector space. The construction of such vectors relies on term frequencies and co-occurrences. These techniques are motivated and extended from a *VSM* designed for *System for the Mechanical Analysis and Retrieval of Text (SMART)* in the 1970s [154][6]. Several decades later, in [10], neural network techniques were introduced in *VSMs* to represent word meanings. Another decade after that, one of the most important word meaning representations, named *Word2vec* (still based on *VSMs* and neural network techniques) was invented [98] [98]. The underlying principle of *Word2vec* is based on an observation that a word's meaning is highly related to the co-occurrences of this word and the words occurring around it within a certain range (the surrounding words are usually called the *context words*). There are two approaches to model the relationships between a word and its context words. The first is *continuous bag-of-word* [97] which predicts the current word based on the context words. The other is *skip-gram* [97] which predicts the context words based on the current word. Most state-of-the-art *DSRs* such as *Doc2vec*, *Sent2Vec* [107], *Phrase2Vec* [7] and *fastText* [15] are extensions of *Word2vec* by taking documents, sentences, phrases and letters as contexts.

The success of *Word2vec*-based semantic representations[7] comes from certain properties that are convenient for computing semantic similarities. First, each word is represented as a vector and is embedded into the same vector space. Then the similarities between words can be conveniently computed by using *cosine similarity*. Second, in [97], the authors show that algebraic operations can be applied to word vectors to generate meaningful patterns. For example, represented by vectors, "Brother − Man + Woman" is close to the vector for "Sister". This property provides a convenient approach to construct a *DSR* based on word vectors.

Word2vec-based semantic representations, however, have a couple of issues. First, training an effective representation model can be challenging. For instance, in [79], the subtleness of training a *Word2vec* model over a *Wikipedia* dump is studied

[6] In *SMART*, the *VSM* designed for it was actually used to represent documents. In other words, each document is represented by a vector. Each dimension of the vector space corresponds to a word occurring in the documents.

[7] Not only *DSRs* but also semantic representations at other levels such as sentences [30].

using a training set with 35 million documents and 2 billion tokens, and there are 6 input parameters to tune. Additionally, in [86] and [50], the authors point out some challenges of applying neural network techniques in *NLP*, and some challenges of parameter optimization in neural-network-based methods in general are identified in [171]. Second, whether *Word2vec* has implied syntactic relations between words in a sentence is still in dispute. We are interested in this issue because syntactic relations is one of the most fundamental ingredients to represent semantics of sentences and documents [125] [90] [66] [27]. In the original work of *Word2vec* [97], the authors state that the second property (using algebraic operations) of *Word2vec* implies that some subtle *"semantic relationships"* between words can be detected. However, firstly, such semantic relationships are merely observations; secondly, such *"semantic relationships"* are illustrated by a couple of intuitive examples as shown above but are never formally defined; and therefore it is unknown if such *"semantic relationships"* in any sense imply syntactic relations. Beyond the original work of *Word2vec*, on one hand, some works such as [141], [140] and [63] suggest that explicitly introducing syntactic structures (e.g. *parse trees* [68]) and equipping *Word2vec* with such structures will lead to more sophisticated solutions to some problems (e.g. *word similarity* [63], and *sentiment analysis* [140] [141] [147]). On the other hand, [80] and [74] show that more effective semantic representation models can be achieved by adjusting the structures of neural networks without introducing any explicit syntactic structures.

Note that there are also some other state-of-the-art works that do not rely on neural network models but are still based on word embeddings, such as *GloVe* [110] and *LexVec* [126]. *GloVe* takes local context windows and co-occurrence information into a weighted least squares regression model. *LexVec* factorizes positive point-wise mutual information matrix using stochastic gradient descent to obtain representation vectors.

1.2.1.2 Syntactic Structure Based Methods

In an alternative category of semantic representations[8], syntactic structures are playing the core role [44]. Specifically, there are two major types of syntactic structures: *dependency relations*[9] and *constituency relations*.

The study of dependency relations was originated from the work of Lucien Tesnière [149], and the syntactic analysis based on dependency relations primarily concentrates on *argument structures* [18]. Intuitively, an *argument structure* describes the logic relationships between words in a sentence, especially between a verb and a collection of words related to the verb conveying information such as *"who did what to whom, where, when and why"* [2]. Dependencies in a sentence can be represented by a *dependency-based parse tree (DBPT)* [68]. An example *DBPT* is shown in Figure 1.1[10]. Typical semantic representation methods focusing on dependency relations include: the work in [49] which is specific to the *Semantic Role Labeling* problem [68]; *UCCA* [1] and *UDS* [161] which are syntactic annotation methods; and *Abstract Meaning Representation (AMR)* [9] which represents a single sentence as a graph consisting of syntactic elements and their relations.

On the other hand, the constituency-relation-based syntactic analysis was primarily contributed by Noam Chomsky [27]. A *constituent* in a sentence is a sequence of words of the sentence, together as a unit in a hierarchical structure (usually a tree structure), carrying a meaning [27], and *constituency relations*[11] are the compositional

[8] In this alternative category, semantic representations are specific to sentences [44], which can be extended to documents.

[9] Note that the term *dependency* in Computational Linguistics is not specific to syntactic dependencies (i.e. dependency relations between words in a sentence specified by syntax). There are also other types of dependencies such as *semantic dependencies* [95]. Different types of dependencies describe relations between words from different perspectives. For example, consider the two sentences: *"Errol sees a dinosaur."* and *"A dinosaur is seen by Errol."* From the syntactic dependency perspective, the words *"Errol"*, *"see"* and *"dinosaur"* are of different syntactic relations whereas, from the semantic dependency perspective, their semantic relations never vary. Here we only consider syntactic dependencies.

[10] Figure 1.1 is generated by *AllenNLP* [48]

[11] Sometimes the term *phrase structure* is used, though the term *constituency relation* is used in a

relations between constituents. Constituents and their relations of a sentence can be represented by a *constituency-based parse tree (CBPT)* [68]. An example *CBPT* is shown in Figure 1.2[12]. Typical semantic representation methods focusing on constituency relations include *CCG* [144] which is a grammar formalism, and the work in [85] which utilizes *CCG* and first order logic to address the *question answering* problem [52] [163]. The syntactic-structure-based semantic representation methods typically reconcile with computational linguistics better than neural-network-based methods, because the semantic representations provided by those methods explicitly reflect syntactic structures, and thus those representations are interpretable in detail[13]. For example, dependency and constituency based parse trees are actually semantic representations for sentences. Figure 1.1 and 1.2 have shown the syntactic details of the representations. Note that, vector representations derived by *Word2vec* or its extensions do not reflect syntactic or semantic details as well as the syntactic-structure-based semantic representations. However, the syntactic-structure-based semantic representations typically do not have forms as convenient as vectors (e.g. *AMR* uses graphs as its representations), and are not immediately applicable to many semantics-oriented problems such as *document semantics comparison* and *document clustering*.

1.2.2 "Phrases"

Learning from the semantics representation methods discussed above, naturally our goal is to design vector-formed *DSRs* that explicitly utilize syntactic structures but without using neural network models. Key to our approaches is to decide which type of syntactic structures should be utilized (i.e. dependency relations or constituency relations). Our choice is motivated by some previous work taking advantage of *n-grams* [68] such as *Phrase2Vec* and a document clustering method named *Suffix Tree*

more general sense.

[12] Figure 1.2 is generated by Stanford CoreNLP [93] on website: http://nlpviz.bpodgursky.com/

[13] With respect to *Word2vec* or its extensions, to date there is no interpretation about what semantics correspond to by each dimension of a representation vector.

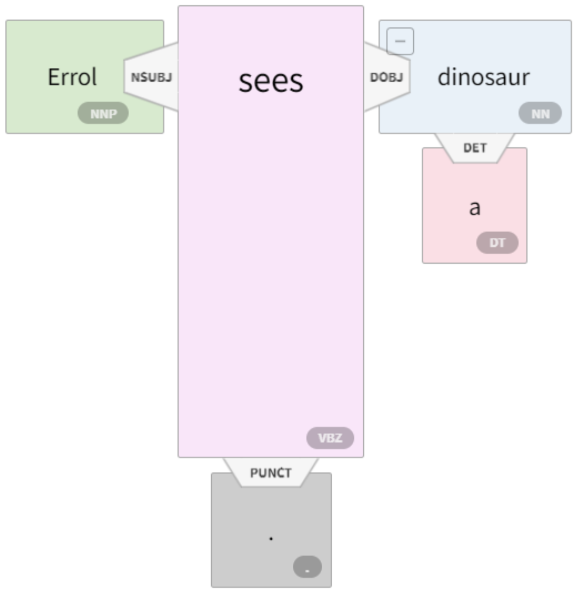


Figure 1.1: An example of *DBPT*. *"See"* is a verb, *"dinosaur"* is the accusative object of the verb, and *"Errol"* is the nominal subject of the verb. These syntactic relations between words are called the *dependency relations*.

Clustering (STC) [168] [167].

Next, we review *STC* to explain how our works are motivated. The core idea of *STC* is to find common *phrases* (i.e. n-grams) between documents by utilizing *Suffix Trees* [158] to represent the documents. Here, a *phrase* in [168] and [167] means a sequence of consecutive words in a sentence. The more common *phrases* there are between two documents, the more semantically similar the document. For example, consider the following three semantically similar sentences:

- *"Cat **ate cheese**."*
- *"Mouse **ate cheese** too."*
- *"Mouse **ate** more **cheese**."*

In the first two sentences, by using *STC*, three common *phrases* can be found: *"ate"*, *"cheese"* and *"ate cheese"*. Then the semantic similarity between the two document can be computed by, for example, *Jaccard similarity*. Particularly, *"ate cheese"* is a typical 2-gram. However, when considering the third sentence, *STC* cannot capture

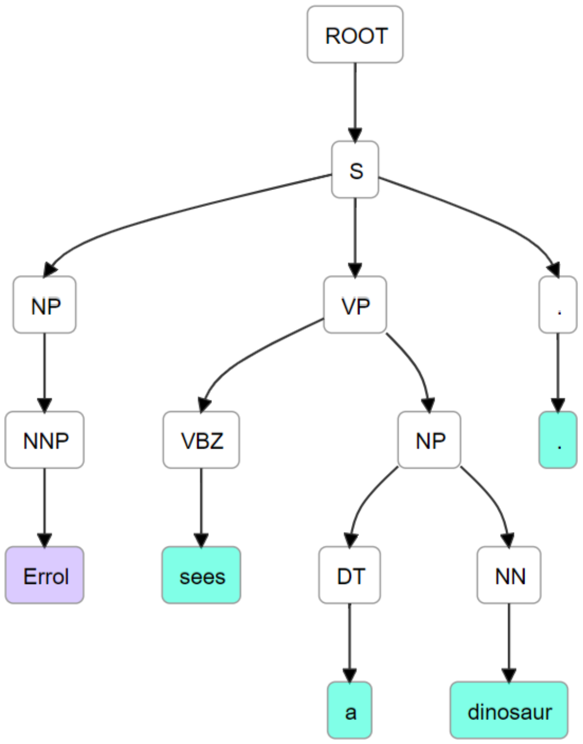


Figure 1.2: An example of *CBPT*. The original sentence can be parsed into a *noun phrase (NP)* (i.e. *"Errol"*) and a *verb phrase (VP)* (i.e. *"sees a dinosaur"*). This *VP* can be further parsed into a *verb* (i.e. *"sees"*) and another *NP* (i.e. *"a dinosaur"*). *NPs*, *VPs*, *verbs* and *nouns* in this example are *constituents*. These compositional relations between the constituents are called the *constituency relations*.

"*ate ... cheese*" which is semantically similar to *"ate cheese"*, because "*ate ... cheese*" is not a consecutive sequence of words. In other words, n-grams can only capture linear relations of words. Thus, *STC* cannot detect the semantic similarities between the third sentence and the first two sentences.

Nonetheless, *STC* suggests the possibility of representing document semantics based on "phrases" (defined in an appropriate way). Thus, motivated by these works, we ask: is it possible to define "phrase" in syntactic structures? We answer this question in the affirmative by defining phrases with syntactic relations. To do this, we observe, in a *CBPT* (to represent the constituency relations in a sentence), that any two words[14] are related by a constituent, and the significance of this syntactic relation

[14] We do not consider *stopwords* (e.g. *"a"*, *"the"* and *"she"*) [19].

is reflected by the path length between the two words in the *CBPT*. Actually this idea coincides with *syntactic n-grams* [137] [138], which is defined as an n-gram that takes its terms in the order of these terms appearing in a parse tree (*CBPT* or *DBPT*). To date, few previous works utilize syntactic n-grams [138], and most of these works only utilize dependency-based syntactic n-grams [137] [136]. To understand why syntactic n-grams are superior to n-grams, we show an example in Figure 1.3. In this example, we use the three example sentences discussed above. Specifically, by using syntactic n-grams, "*ate* ... *cheese*" in the third sentence can be captured as a semantic similar phrase to "*ate cheese*" because these two phrases have the same words and also the same syntactic relation.

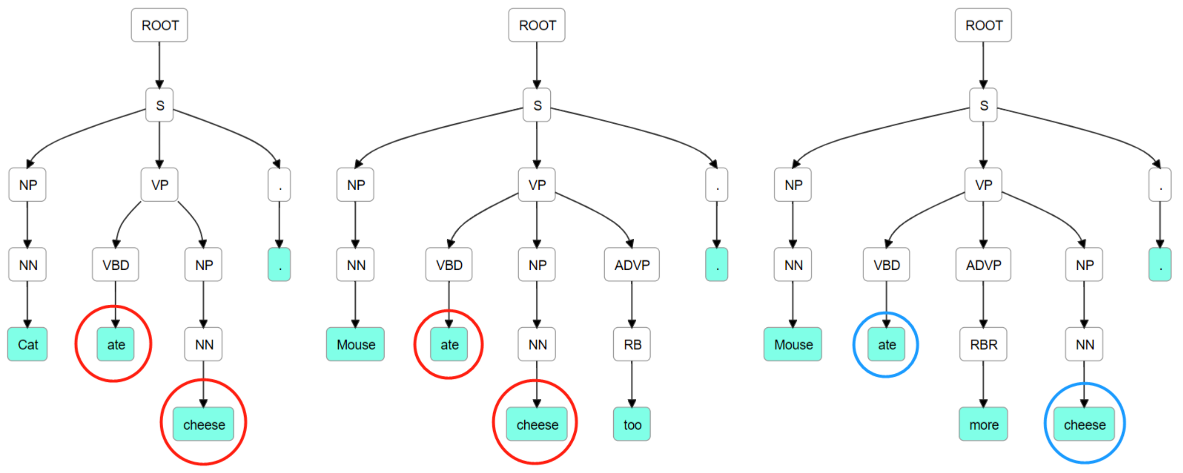


Figure 1.3: N-grams can only capture "*ate cheese*" that occurs in the first two sentences but not "*ate* ... *cheese*" in the last sentence. Syntactic n-grams can capture them all.

However, syntactic n-grams have two issues: first, a syntactic n-gram cannot reflect the significance of this n-gram in representing semantics; and second, constituency-based syntactic n-grams are sensitive to the word order.

To show the first issue, consider the example shown in Figure 1.4. The figure shows the *CBPT* of the sentence: *"We eat pizza when we watch a movie."* In this *CBPT*, for example, *"watch"* and *"movie"* are related by a *VP* (marked by the red box); meanwhile, *"eat"* and *"movie"* are also related by another *VP* (marked by the blue box). From the sentence semantics point of view, obviously "*watch*" and "*movie*", together as a pair, carry a more significant meaning than that carried by the pair *"eat"* and *"movie"*. This difference in significance of meaning is reflected by difference of path lengths between the two pairs of words (i.e. the path length between *"watch"* and *"movie"* is 5 which is shorter than 8, the path length between *"eat"* and *"movie"*). Thus, the shorter the path length, the more significant the syntactic re-

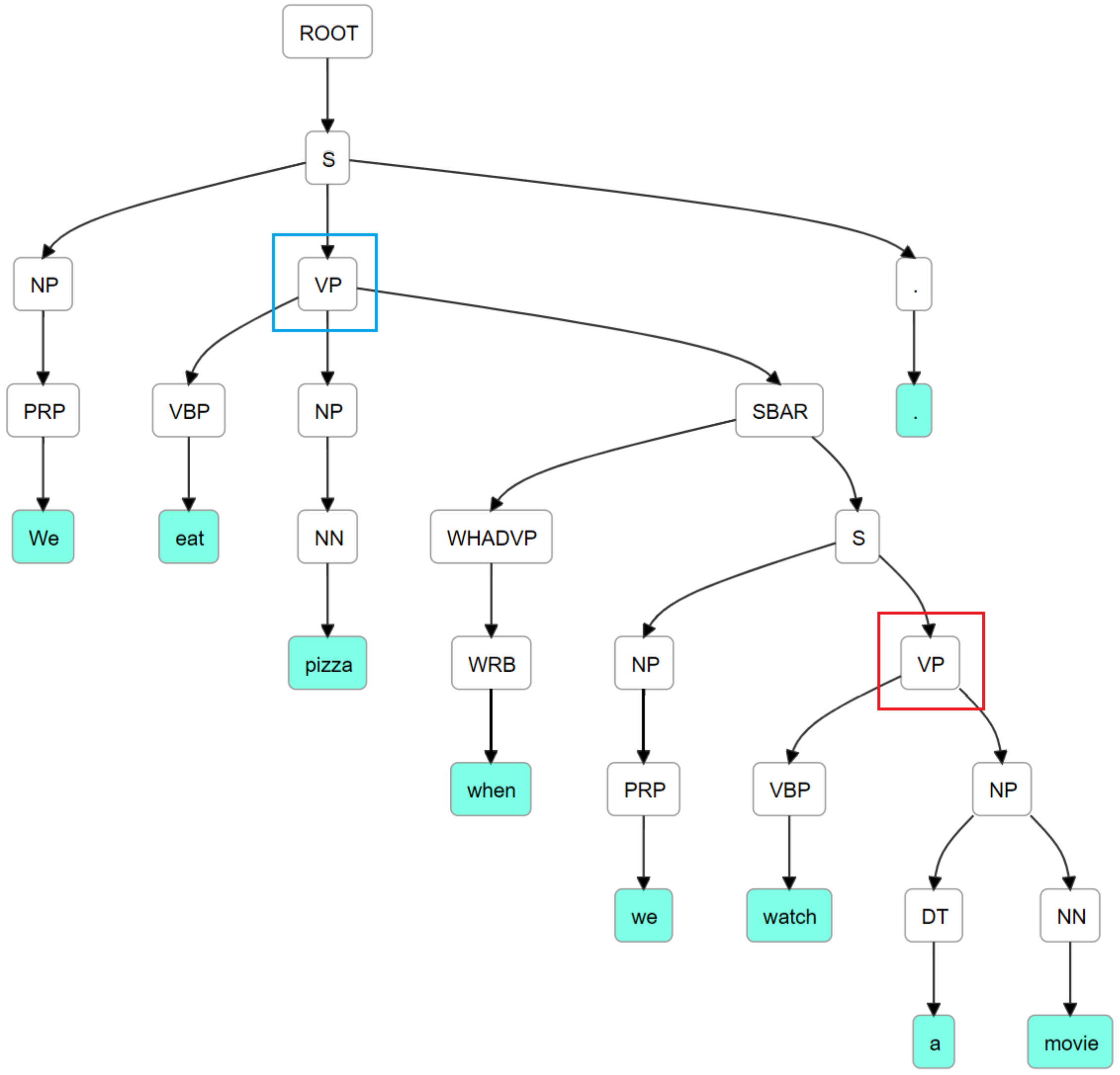


Figure 1.4: An example to show that the path length of two words in a sentence reflects the significance of their syntactic relation.

lation. In addition, in linguistics, theories agree that in general syntax follows from semantics [162] [162]. Thus, naturally the significance of a syntactic relation in general implies the significance of the corresponding semantics.

To show the second issue, consider the following two sentences:

- "*Ben is a good* ***bike rider****.*"
- "*Ben is* ***riding*** *a* ***bike****.*"

The phrase "*bike rider*" in the first sentence is semantically similar to the phrase "*riding . . . bike*" in the second sentence. The *CBPTs* of these two sentences and the syntactic n-grams of these two phrases are shown in Figure 1.5. It is straightforward to see that, taking the word orders into consideration, the two syntactic n-grams will not be considered as similar, though from the semantics perspective they are semantically similar. Thus, if the word order is relaxed, then syntactic n-grams can capture more semantic similar phrases.

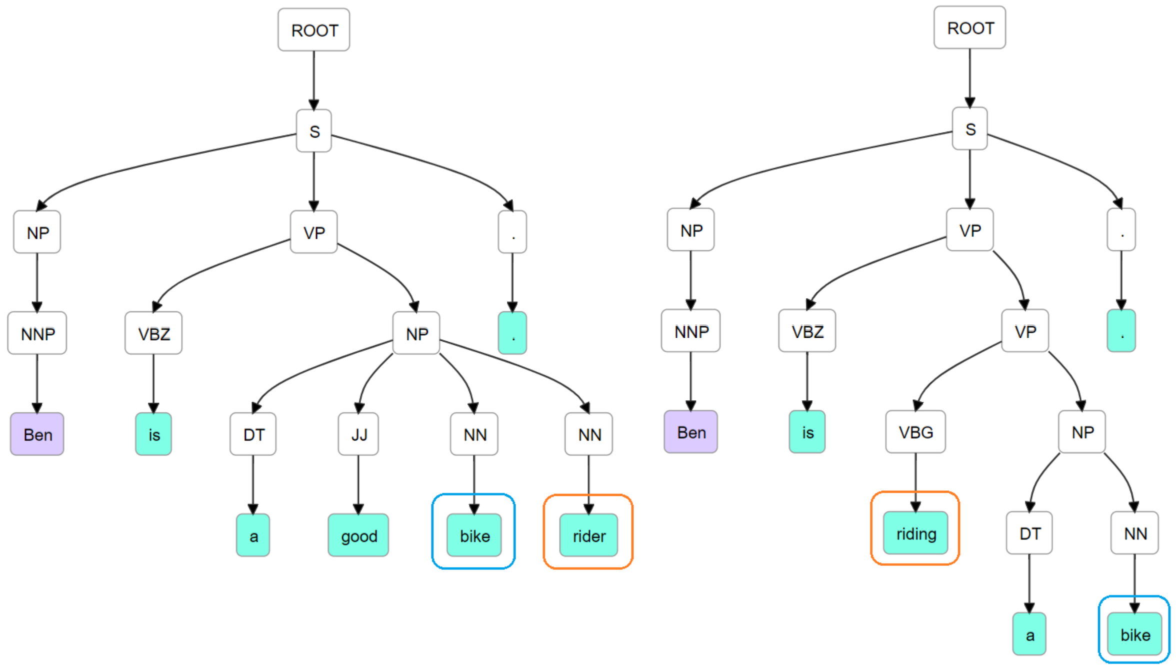


Figure 1.5: An example to show that the path length of two words in a sentence reflects the significance of their syntactic relation.

Based on these observations and discussions, we propose a new definition of "phrase", named **generalized phrase (GP)**.

Def: Generalized Phrase (GP)

A **generalized phrase** is a minimal non-empty subtree of the *CBPT* containing at most two leaves (i.e. two words)[15]. The leaves are considered as orderless. The significance of the relatedness between the leaves is computed by using the path length between them. The shorter the path length, the more significant the *GP*. If only one word is contained in a *GP*, then the path length of this *GP* is set to 1.

By the definition, *GPs*, firstly, capture the words in a sentence; and second, capture the constituency relations between words. In the example shown in Figure 1.4

Motivated by *GP* and all discussions above, this dissertation is aimed at the following question: **Is it possible to design *DSRs* based on *GPs* without using neural network models?**

To answer this question, we begin by setting aside the *DSR* design issue and studying the effectiveness of *GPs* in reflecting document semantics. The idea is: For two documents, we extract semantically similar *GPs*[16] between the document, then we compute a similarity value based on the extracted *GPs*. Then if this similarity value is in accordance with a predetermined similarity (e.g. by human judgments) between the two documents, it will be sufficient to show the effectiveness of *GPs* in reflecting document semantics. This idea motivates our first major contribution, *DSCTP*, which is described in Chapter 2.

Note that having established the effectiveness of *GPs*, we turn to the design of *DSRs*. *DSCTP* is a method for document semantics comparison rather than an immediate *DSR*. Specifically, *DSCTP* can be utilized to extract *GPs*; however, we also need to answer how to construct a semantic representation for a single document based on the *GPs* extracted from that document. In addition, we require that our

[15] *Stopwords* [19] should always be removed and will never appear in any *GP*.

[16] We formally define the term *"semantically similar"* of *GPs* in Chapter 2. Up to this point, this term can be intuitively understood as "*carring similar meanings*".

DSRs should be more intuitive and straightforward than *NNWE*-based methods, and that they use vector forms. Two *DSRs*: *APV* and *GPGS* are described and evaluated in Chapters 3, 4 and 5.

As an application of our work, we consider a topic in *social dynamics* [41] that relies on the comparison of documents [112]. Since more than a decade ago, it has been observed that specific communities (e.g. *business*, *travel* and *free time*) [96] exist in social networks. This observation implies that if we consider the message history (i.e. the collection of messages within a period) issued by a user in a social network as a document, then when the user sees a message m issued by another user, the more the message m is semantically similar to its message history, the more likely this user propagates the message m. Thus, *DSCTP* can be used to compare two users with respect to their issued messages, and also *APV* and *GPGS* can be used to represent the semantics of a message and the semantics of the message history. Thereby, our research on message propagation in social networks can utilize our document semantics comparison methods and *DSRs*.

Chapter 2

DOCUMENT COMPARISON

2.1 Introduction

Following the motivations discussed in Chapter 1, we explore the design of *DSRs* based on *GPs*. We begin by studying the effectiveness of *GPs* in reflecting document semantics. Specifically, we examine if *GPs* are effective ingredients when they are utilized in pairwise document semantics comparisons.

Comparing document semantics is one of the most fundamental tasks in both *Natural Language Processing (NLP)* and *Information Retrieval (IR)* [51]. In this chapter, the problem of pairwise document comparison is formulated as follows, named the **Document Semantics Comparison (DSC) Problem**:

Document Semantics Comparison (DSC) Problem:

Input:

Two *documents*[1] in English consisting of one or more sentences.

Seek:

A single real value to reflect the *semantic*[2] similarity between the two documents.

[1] Here, the term "*document*" follows the description of documents in Chapter 1.

[2] The term "*semantic*" follows the explanation of semantics in Chapter 1.

To address the *DSC* problem, we present a new *training-free*[3] document semantics comparison framework named **DSCTP**. *DSCTP* is called a *framework* because it is designed in a modular style[4]. In addition, *DSCTP* takes *GPs* as principal ingredients, and introduces techniques from *algebraic topology* [100] [59] as primary tools to build the framework. Note that *DSCTP* is the first application of algebraic topology techniques in addressing the *DSC* problem, and is also an interesting exploration of *NLP* scenarios in which such techniques can play a critical role[5].

The intuition behind *DSCTP* is straightforward: the larger the number of semantically similar[6] *GPs* across the two documents, the more semantically similar the two documents. Motivated by the intuition, the core idea of *DSCTP* is: firstly, extract semantically similar *GPs* across the two documents; secondly, computing a document semantic similarity score based on these *GPs*.

To evaluate *DSCTP*, a series of experiments are conducted comparing *DSCTP* with a collection of state-of-the-art methods addressing the *DSC* problem including *Word Mover's Distance (WMD)* [77], *Doc2Vec* [80], *NASARI* [21], *GloVe* [110], *fastText* [15], *LexVec* [126] and *Sent2Vec* [107]. Four popular datasets are used in our experiments: *Lee50* [82], *Li65* [89], *STS2017* [23] and *SICK* [94]. Each contains a set of documents and a full set of pairwise document similarity scores determined by humans. On these four datasets, every method mentioned above, including *DSCTP*, is applied to all document pairs, producing either a semantic similarity score or distance for each pair. These results are then compared against the provided human judgments to evaluate how well the methods perform in accordance with human judges. The

[3] "*Training-free*" means that *DSCTP* does not rely on any training set for model building. Note that *DSCTP* does require a lexical semantic similarity method (e.g. cosine similarity on *NASARI* [20] word vectors), and some lexical semantic similarity methods do rely on training sets. However, *DSCTP* is not specific to any lexical semantic similarity method, thus the training set required by a lexical semantic similarity method is not counted as a requirement of *DSCTP*.

[4] The framework is explained in Section 2.4.

[5] A couple of existing *NLP* applications of algebraic topology techniques are reviewed in Section 2.3.

[6] "*Semantically similar*" is explained later in this chapter.

experimental results show that *DSCTP* outperforms all other methods.

Before the official introduction of *DSCTP*, we briefly review some previous work and provide some theoretical background for *algebraic topology* in the following two sections.

2.2 Related Work

A set of state-of-the-art semantic representation methods have been discussed in Section 1.2 including the particular methods that we will compare in this chapter: *Doc2Vec* [80], *NASARI* [21], *GloVe* [110], *fastText* [15], *LexVec* [126] and *Sent2Vec* [107]. These particular methods are word-embedding-based, and thus by utilizing these methods, a document can be represented as a real-valued vector. Then, by using cosine similarity, the semantic similarity between two documents can be computed.

WMD [77] is a distance method between two documents based on *Word2Vec* [97]. *WMD* measures the dissimilarity between two documents by computing the minimum distance that the embedded words[7] of one document need to be "moved" to the positions of the embedded words of another document. The experimental results provided in [77] show that *WMD* outperforms a set of classic and state-of-the-art document representations and distances including *Bag-of-Words (BOW)* [57], *TF-IDF* [92] [143], *BM25 Okapi* [121], *Latent Semantic Indexing (LSI)* [38], *Latent Dirichlet Allocation (LDA)* [14], *mSDA Marginalized Stacked Denoising Autoencoder* [26], and *CCG Componential Counting Grid* [111].

In our *DSCTP*, a lexical similarity method is required, but *DSCTP* is independent of the choice of lexical similarity method. To evaluate the performance of *DSCTP*, in our experiments, we utilize two different types of lexical similarity methods. One is *NASARI* [21] equipped with cosine similarity. The other is *ADW* [114] which is based on semantic networks [142] instead of word embeddings. *NASARI* has been discussed

[7] Recall that *Word2Vec* is a word-embedding-based method.

earlier. Here, we briefly review *ADW* [114]. *ADW* works by utilizing *semantic networks*[8]. *ADW* proposes a concept, *semantic signature of lexical term*, which is defined as the multinomial distribution generated from random walks over a semantic network (e.g. *WordNet* [109]). To obtain such a semantic signature of a word in a sentence, *ADW* requires the *sense*[9] of this word. To satisfy this requirement, *ADW* also provides a method to disambiguate word senses. The core idea of this method is that, for two words, the distances or similarities of all pairs of senses of those words are examined, and the best distance or similarity suggests the pair of senses that is utilized. They also propose a vector comparison method named *Weighted overlap* similarity that is used to compare two word signatures. *Weighted overlap* similarity is defined by

$$\frac{\sum_{i=1}^{N}(r_i^1 + r_i^2)^{-1}}{\sum_{i=1}^{N}(2i)^{-1}}$$

where r_i^j denotes the rank of the value at the *ith* dimension of vector j. The denominator is a normalizer (i.e. the maximum value of the numerator) so that the resulting similarity is between 0 and 1.

Finally, we note that there are several problems which are related to the *DSC* problem but are quite distinct. First, the *word-level semantics comparison* [67] and *sentence-level semantics comparison* [5] problems concentrate on semantic similarities of words and sentences but not documents. Second, the *sentiment analysis* problem [108] aims at extracting and comparing the sentiments expressed by documents. This may involve figures of speech that are not taken into consideration in the *DSC* problem. Third, the *topic modeling* problem [13] whose core task is extracting *topics* from documents. *Topics* are typically represented by a small set of words. Note that

[8] Specifically, *ADW* utilizes *WorkNet* [109]. Other semantic networks such as *BabelNet* [103] are also possible to be adapted to *ADW*; however, such adaptations have not been published.

[9] A word, in different contexts, usually has different meanings. A *sense* of a word is one of such meanings of this word.

this problem is not necessarily related to semantic comparisons of documents, though addressing the *DSC* problem first may help with extracting *topics* from documents. Fourth, the *document clustering* [145] problem which groups documents with similar semantics. Solving the *DSC* problem combined with a clustering method (e.g. *spectral clustering* [157]) may provide an effective solution to this problem[10]. Finally, the *document classification* [16] problem which classifies a given document to a predefined semantic group. Since comparing semantics between documents would be a necessary step in solving this problem, then naturally a solution to the *DSC* problem would be helpful for classifying documents.

2.3 Algebraic Topology on Graphs in a Nutshell

Algebraic topology, the major tool used by *DSCTP*, is a large and complex topic. For the convenience of readers, some minimum background and concepts of *algebraic topology* are provided in this section focusing on fundamentals of *homology theory* [100] and *topological persistence* [42] which are the major concepts used by *DSCTP*. In this brief view, we assume readers are familiar with some algebraic topology terms. For further details, readers are referred to [100], [42] and [59].

Intuitively, the major task of *homology theory* is to describe "*holes*" in a geometric space from an algebraic perspective. As an intuitive analog, one can imagine an MRI scan over a solid ball with a void inside. Relative to the MRI slices, this void will emerge at some time point and vanish at some other time point. The void is the "*hole*", and the course of the hole emerging and vanishing is what *persistence* reflects. A formal description is provided below.

We start with the most important concept for formulating geometric spaces from the algebraic perspective, namely **abstract simplicial complex**.

[10] In Chapter 5, we will show that integrating *DSCTP* with spectral clustering outperforms state-of-the-art methods.

Def: Abstract Simplicial Complex

An **abstract simplicial complex** is a collection $\mathbb{S}$ of finite sets, such that if A is an element of $\mathbb{S}$, so is every nonempty subset of A.

Based on this core concept, we review a set of concepts.

Def: p-Simplex

In an abstract simplicial complex $\mathbb{S}$, the element A is called a **simplex** of $\mathbb{S}$, and the dimension p of A is one less than the cardinality of $\mathbb{S}$. A **p-simplex** is a p-dimensional **simplex**.

Abstract simplicial complex and p-simplex are not necessarily corresponding to geometric objects. Instead, they are algebraic objects. However, in an intuitive way, it would be helpful to understand these concepts from a geometric perspective. For example, a line segment can be considered as a 1-*simplex*, and a triangle convex hull can be considered as a 2-*simplex*. Additionally, it is important to note that an abstract simplcial complex can correspond to a geometric object which is a **simplicial complex** [100], and this simplicial complex is called a **geometric realization** of the abstract simplcial complex. In a simplcial complex, a p-simplex is **oriented** [100] and can be denoted by a sequence of vertices: $\sigma_p = [v_0, \ldots, v_p]$.

Def: p-Chain

A p-**chain** is a *formal sum*[11] of a set of p-simplices, written as $\sum\limits_k \alpha_k \sigma_k$, where α_k is a coefficient in the ground field $\mathbb{F}$ and σ_k is a p-*simplex.*

[11] Loosely speaking, a *formal sum* can be understood as a generalization of *weighted sum* of numbers. The weights in a formal sum typically are integers, especially in *group theory.* The elements being summed, in our context, are p-simplices.

Graphs are also abstract simplicial complexes, and in this dissertation, we only consider dimension 1. In other words, we do not consider surfaces or higher dimensional geometric objects.

Def: Boundary Operator, p-Cycle and p-Boundary

A **boundary operator** is a *homomorphism* which maps *p-chains* to $(p-1)$-*boundaries*, denoted by ∂_p such that $\partial_p \sigma_p = \sum_{i=0}^{p} (-1)^{-1} [v_0, \ldots, \hat{v_i}, \ldots, v_p]$ where $\hat{v_i}$ means that the vertex v_i is to be removed from the sequence. A **p-cycle** is a *p-chain* to which applying the *p-boundary-operator* will return zero. A **p-boundary** is an image of a $(p+1)$-*chain* to which applying the $(p+1)$-*boundary-operator*.

For example, given a triangular space which is a 2-chain, a 2-boundary-operator takes this convex hull and returns the triangle consisting of three line segments without the interior of the space. This triangle is a 1-chain. Then, this 1-chain is called the 1-boundary of the convex hull. If we apply a 1-boundary-operator to this 1-boundary, then the result is zero. Thus, this 1-boundary is also a 1-cycle. Moreover, a property of boundary operators is implied from this example, and is stated as follows:

Lemma: Chain Complex Property

$\partial_p \circ \partial_{p+1} = 0$

Based on the concepts of p-boundary-operator, p-chain, p-cycle and p-boundary, we are ready to review the concepts of *p-homology-class* and *p-homology-group*. An example of these concepts is shown in Figure 2.1.

Def: Homology Class, Homology Group

A **p-homology-class** is a set of p-cycles equivalent to one another, and the equivalence relation is defined such that if two *p-cycles* c_p^i and c_p^j are equivalent then $c_p^i - c_p^j$ is a *p-boundary*.

A p**-homology-group** is the set of p-homology-classes computed from an abstract simplicial complex. In a formal way, the *p-homology-group* is defined as $\mathcal{H}_p = \mathcal{Z}_p/\mathcal{B}_p$, where $\mathcal{Z}_p$ is the *p-cycle group*, and $\mathcal{B}_p$ is the *p-boundary group*. Equivalently, the *p-homology-group* is defined as $\mathcal{H}_p = Ker(\partial_p)/Im(\partial_{p+1})$.

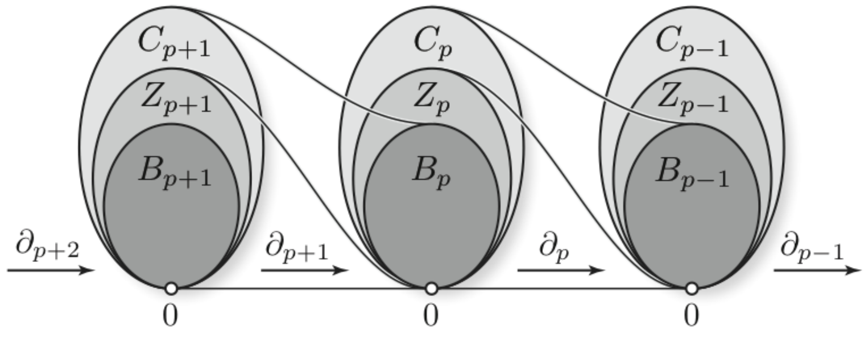


Figure 2.1: Chain complexes & Homology groups. ∂_p's are boundary operators, C_p's are chain groups, Z_p's are cycle groups, and B_p's are boundary groups. The *kernel* of the p-boundary-operator is the p-cycle group, and the *image* of the $(p+1)$-boundary-operator is the p-boundary group. The p-homology-group is a *quotient group*: Z_p modulo B_p.

On graphs, 1-homology-groups can be obtained by computing a *minimum cycle basis* [70] [71], and each 1-homology-class corresponds to a *minimum basic cycle.*

Next, we review a set of concepts in **topological persistence** which is an extension to (co)homology theory.

Def: Filtration, Birth, Death, Lifespan & Barcode

A **filtration** of an abstract simplicial complex K is a nested subsequence of abstract simplicial complexes: $\emptyset = K^0 \subseteq K^1 \subseteq \cdots \subseteq K^m = K$, and let $K^i = K^m$ for all $i \geq m$. The indices $0, \ldots, m$ are called the **filtration values**. Each simplex in an abstract simplicial complex is assigned a filtration value.

The **birth** of a p-homology-class is the lowest filtration value at which a representative p-cycle of this p-homology-class emerges.

The **death** of a p-homology-class is the lowest filtration value at which a $p+1$-cycle containing the exact set of vertices of any of this p-homology-class's p-cycles emerge (N.B. this moment can be infinity). In our cases, for convenience of computation, infinity is substituted by a predetermined value.

The **Lifespan** of a p-homology-class is the distance between its birth and death.

A visualization of the set of p-homology-classes with birth-death pairs computed from an abstract simplicial complex with a filtration is called a **persistence diagram**. In a persistence diagram, each p-homology-class with its birth-death pair is presented as a line segment starting from the birth point and ending at the death point. This line segment is called the **barcode** for the corresponding p-homology-class.

An example illustrating the concepts of abstract simplicial complex, homology groups and topological persistence on graphs is shown in Figures 2.2, 2.3 and 2.4.

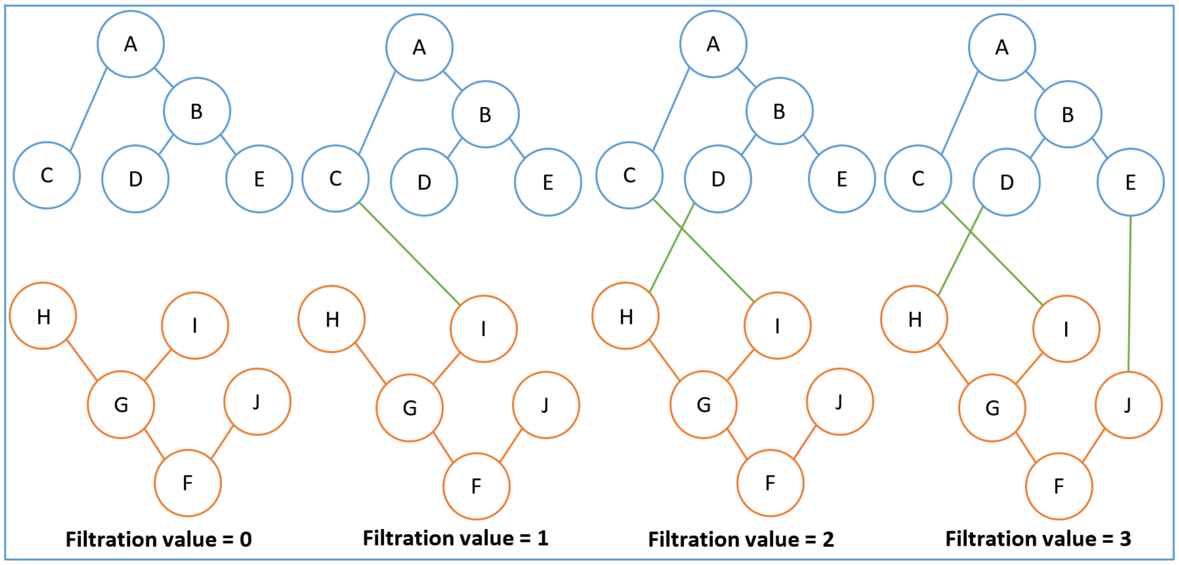


Figure 2.2: Two trees (i.e. the blue one and the orange one) as two abstract simplicial complexes appear at the moment 0. Initially, all simplices contained in the two trees are assigned filtration value 0. At the moment 1, an edge between C and I is created and is assigned filtration value 1. Similarly, at the moments 2 and 3, another two edges are created, and they are assigned corresponding filtration values.

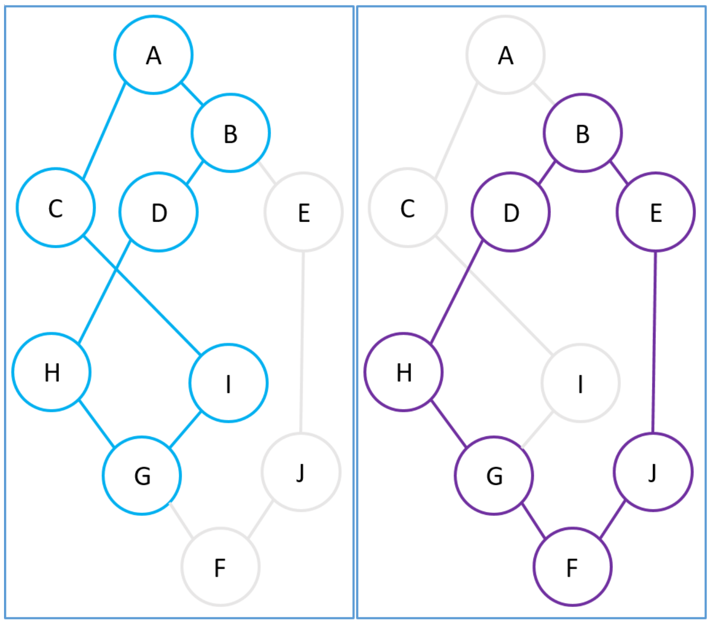


Figure 2.3: Based on the abstract simplicial complex at the moment 3 in Figure 2.2, the homology group is computed, and there are two homology classes appearing as two *minimum basic cycles* (i.e. the blue cycle to the left and the purple cycle to the right).

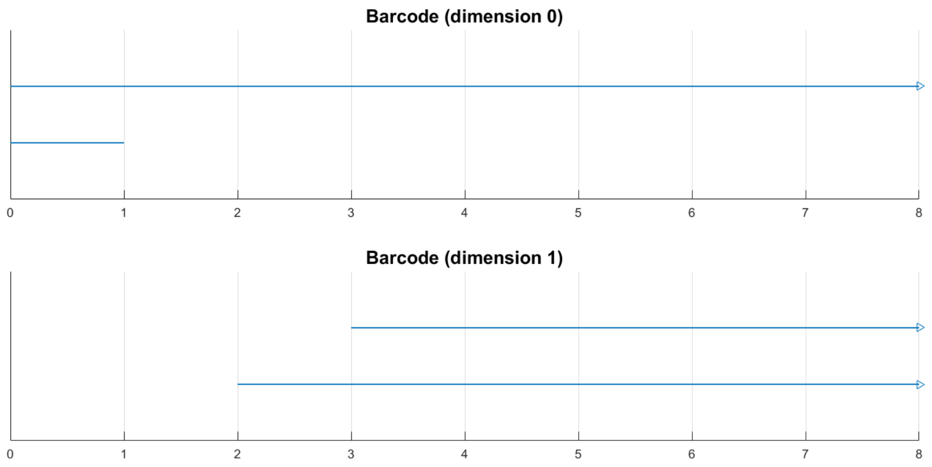


Figure 2.4: Based on the abstract simplicial complex and the filtration value assignment shown in Figure 2.2, the topological persistence for the homology group shown in Figure 2.3 is computed. The corresponding barcode is shown in this figure. The 1-dimensional barcode shows that initially there are two connected components which are the two trees. From the moment 1 the two components are connected, and thus only one connected component appears after that. The 2-dimensional barcode shows that initially there is no cycle in the abstract simplicial complex. At the moment 2, one cycle emerges, and that is the blue cycle shown in Figure 2.3, and at the moment 3, another cycle emerges, and that is the purple cycle. These two cycles then persist until infinity.

2.4 Document Semantics Comparison Based on Topological Persistence (DSCTP)

In this section, we explain *DSCTP* in detail. Recall that, to score the similarity between two documents, *DSCTP* has two major phases: first, extract semantically similar *GPs* across the documents; second, compute the document similarity score based on the extracted *GPs*. Because of the sophistication of *DSCTP*, here we concisely preview these two phases, and then we elaborate on the details of *DSCTP*.

Phase 1: Extract Semantically Similar GPs

Syntactic relations of words in each sentence of the two documents and word semantic relations across the two documents are required in this phase. Syntactic relations are obtained from constituency-based parse trees (i.e. *CBPTs*), and word semantic relations are obtained from lexical similarities. A graph G as a natural structure to express these relations is constructed. Thereafter, since a graph is also an abstract simplicial complex, a 1-homology-group is computed over this graph G, and a pair of semantically similar *GPs* will be extracted from each 1-homology-class in the 1-homology-group.

Phase 2: Measure Document Similarity Score

The similarity score for the two documents is obtained from topological persistence computed over the 1-homology-group of graph G. As a key factor in computing topological persistence, the filtration values assigned to the graphs in the first phase are determined by lexical similarities. In the resulting topological persistence, the lifespan of each 1-homology-class then reflects the *semantic similarity* between the two corresponding *GPs*. Also, for the pair of *GPs* in each 1-homology-class, we assign a weight which reflects their significance in reflecting the semantic similarity between the documents. This weight is determined by the path lengths of the *GPs*. Finally, the document semantic similarity score is computed by a weighted sum over the semantic similarities of the *GP* pairs associated with all 1-homology-classes.

To detail the framework, *DSCTP*, we expand the two major phases into eight stages which are presented as a pipeline. These stages are given in **Algorithm 2.1**, and explained as necessary in the rest of this section. This pipeline is shown particularly in **Figure 2.5**.

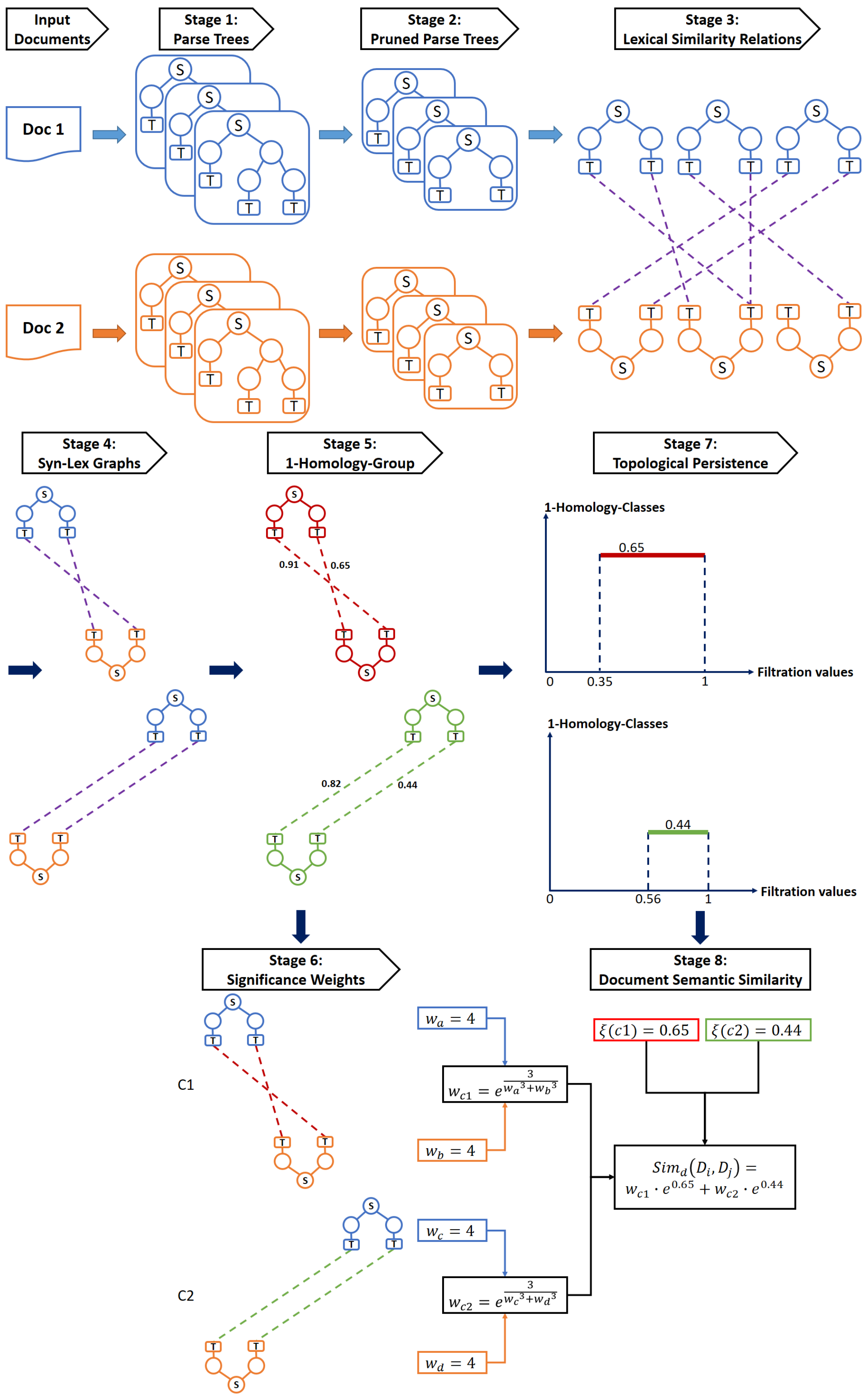

Input Documents
Stage 1: Parse Trees
Stage 2: Pruned Parse Trees
Stage 3: Lexical Similarity Relations
Doc 1
Doc 2
S
T
Stage 4: Syn-Lex Graphs
Stage 5: 1-Homology-Group
Stage 7: Topological Persistence
0.91
0.65
0.82
0.44
1-Homology-Classes
0.65
0
0.35
1
Filtration values
1-Homology-Classes
0.44
0
0.56
1
Filtration values
Stage 6: Significance Weights
Stage 8: Document Semantic Similarity
C1
C2
$w_a = 4$
$w_{c1} = e^{\sqrt[3]{w_a{}^3 + w_b{}^3}}$
$w_b = 4$
$w_c = 4$
$w_{c2} = e^{\sqrt[3]{w_c{}^3 + w_d{}^3}}$
$w_d = 4$
$\xi(c1) = 0.65$
$\xi(c2) = 0.44$
$Sim_d(D_i, D_j) = w_{c1} \cdot e^{0.65} + w_{c2} \cdot e^{0.44}$

Figure 2.5: The pipeline of *DSCTP*.

Algorithm 2.1: DSCTP Framework

Given:

- Two documents, D_i and D_j.
- A threshold for lexical similarities, θ_w.
- A *stopword* [119][12]list, $\mathbb{W}_s$[13].
- A set of *name entities* [102][14]in consideration, $\mathbb{N}$.
- A set of *part-of-speech (POS)* [68] tags[15]in consideration, $\mathbb{P}$.

Seek:

A real value reflecting the semantic similarity between D_i and D_j, denoted by $Sim_d(D_i, D_j)$.

Stage 1: Parse Trees

For each document D_i, and for each sentence $S_{ik} \in D_i$, where k indexes the sentences, compute a *CBPT* for S_{ik}, denoted by T_{ik}.

Stage 2: Pruned Parse Trees

For each *CBPT* T_{ik}, pruned T_{ik} following a set of specific rules primarily based on $\mathbb{W}_s$, $\mathbb{N}$ and $\mathbb{P}$ (The tree pruning algorithm is shown in **Algorithm 2.2**). Let $\hat{T_{ik}}$ denote the pruned tree.

Stage 3: Lexical Similarity Relations

For each pruned tree pair $(\hat{T_{ik}}, \hat{T_{jh}})$, where $\hat{T_{ik}}$ corresponds to $S_{ik} \in D_i$ and $\hat{T_{jh}}$ corresponds to $S_{jh} \in D_j$, the word pairs (denoted by $\{(t^a_{ik}, t^b_{jh})\}$, where $t^a_{ik} \in \hat{T_{ik}}$ and $t^b_{jh} \in \hat{T_{jh}}$) whose lexical similarities (denoted by $Sim_w(t^a_{ik}, t^b_{jh}) \in [0, 1]$) that are greater than the threshold θ_w are identified.

Stage 4: Syn-Lex Graphs

For each pair of pruned trees $\hat{T_{ik}}$ and $\hat{T_{jh}}$, taking the identified word pairs from **Stage 3**, construct an undirected and weighted graph $\mathcal{G}_{ik,jh}$ on the union of $\hat{T_{ik}}$ and $\hat{T_{jh}}$ and add edges into $\mathcal{G}_{ik,jh}$ induced by the identified word pairs. Additionally, the weights on the edges induced by the identified word pairs are set to w_∞ which denotes a large-enough value; meanwhile, the weights on the edges in $\hat{T_{ik}}$ and $\hat{T_{jh}}$ are set to 1. This graph is called a **Syn-Lex graph** for $\hat{T_{ik}}$ and $\hat{T_{jh}}$.

Stage 5: 1-Homology-Group

For each *Syn-Lex* graph $\mathcal{G}_{ik,jh}$ over k and h, compute a *minimum weight cycle basis (MWCB)* [71], and then, based on this basis, obtain a 1-homology-group, denoted by $\mathcal{H}_{ik,jh}$, which consists of a set of 1-homology-classes. Each 1-homology-class contains two semantically similar *GPs* that belong to D_i and D_j respectively.

Stage 6: Significance Weights

For each 1-homology-class obtained in **Stage 5** (i.e. a *minimum basic cycle*) $c_l \in \mathcal{H}_{ik,jh}$, compute a weight for c_l based on the path lengths of the two *GPs* contained in c_l. This is called the **significance weight**, and is denoted by w_{c_l}. The significance weight measures the significance of these two *GPs* in reflecting the semantic similarity of D_i and D_j.

Stage 7: Topological Persistence

Compute the topological persistence of $\mathcal{H}_{ik,jh}$. Specifically, for each 1-homology-class obtained in **Stage 5**, $c_l \in \mathcal{H}_{ik,jh}$, do the following two steps:

First, for each edge $(v_a, v_b) \in c_l$, let

$$w_{ab} = \begin{cases} Sim_w(v_a, v_b) & \text{if } v_a,\ v_b \text{ are in different parse trees} \\ 1 & \text{otherwise} \end{cases}$$

where $Sim_w(v_a, v_b) \in [0, 1]$;

Second, let $f_{ab} = 1 - w_{ab}$ is the *filtration value* of (v_a, v_b); the desired *lifespan* (i.e. the difference between the *death* and the *birth*) of c_l, denoted by $\xi(c_l)$, is computed as

$$\xi(c_l) = \min_{(v_a, v_b) \in c_l} f_{ab}$$

.

Stage 8: Document Semantic Similarity

Compute the semantics similarity between D_i and D_j as follows:

$$Sim_d(D_i, D_j) = \sum_{k,h} \sum_{c_l \in \mathcal{H}_{ik,jh}} w_{c_l} \cdot e^{\xi(c_l)}$$

Discussions on Algorithm 2.1:

The two major phases of *DSCTP* have been detailed in the eight stages in **Algorithm 2.1**. **Stage 1** to **Stage 5** belong to the first phase concentrating on extracting semantically similar *GPs* across documents. **Stage 6** to **Stage 8** belong to the second phase concentrating on computing document semantic similarity based on the extracted *GPs*. Below we further discuss the technical details and issues at each stage, if any.

The running time for this stage is bounded (though not tightly) by the running time of the *shift-reduce* parsing which in our cases is a reasonable time. The running time of **Algorithm 2.1** is not dominated by this stage.

Stage 2: The purpose of this stage is to simplify the parse trees so that only a minimal amount of necessary constituent elements will be preserved for the document

[12] The subscript s denotes "stopwords".

[13] *Stopwords* are words that have minor contributions to the semantics of a sentence (e.g. "*the*" and "*a*").

[14] *Name entities* mark tokens of special types such as human names and numbers.

[15] *POS* tags are used to mark the types of constituents (e.g. *NN* as in *noun phrase*).

semantics comparison. The main benefit of doing so is that the syntactic structures of sentences become more compact in pruned trees, which will further benefit the time efficiency of **Stage 5** (i.e. computing 1-homology-groups).

The pruning is practiced from three aspects: *stopwords*, *name entities* and *POS* tags. Typically, *stopwords* should be removed because they do not significantly contribute to the semantics of a sentence. In addition, as a part of the modular design of *DSCTP*, it allows users of *DSCTP* to specify the *name entities* and *POS* tags in consideration. For example, words marked as *organization* (a type of name entity) can be more useful than those marked as *person*. When "*UPS*" occurs in a sentence, and "*Fedex*" occurs in another sentence, it is reasonable to infer that the semantics of these two sentences are likely to have concerns about mailing, whereas "*Alice*" and "*Bob*" as in human names may not provide such inference. For another example, consider the sentence "*Errol sees an unbelievably gorgeous dinosaur.*" In this sentence, "*unbelievably gorgeous*" is an *adjective phrase* corresponding to the *POS* tag *ADJP*. It is straightforward to see that, if this phrase is removed from the sentence, the core semantics of this sentence actually would not change significantly. This observation implies that some types of constituents are not as useful as others, under some contexts, in expressing sentence semantics. Thus, for users of *DSCTP* who are assuredly aware of the contexts under which the sentences are processed, customizing the set of *name entities* and the set of *POS* tags in consideration will help simplify the syntactic structures of sentences. The pruning algorithm is given in **Algorithm 2.2**, and an example is shown in Figure 2.6.

The running time of **Algorithm 2.2** is $O(N)$, where N is the number of words in the sentence on which the input parse tree is constructed.

Algorithm 2.2: Parse Tree Pruning

Given:

- A *CBPT* T.

- A *stopword* list, $\mathbb{W}_s$.
- A set of *name entities* in consideration, $\mathbb{N}$.
- A set of *POS* tags in consideration, $\mathbb{P}$.

Seek:

A pruned tree.

```
IF $T$ is a single-node tree, and the only node is denoted by $t_0$:
    IF $t_0 \in \mathbb{W}_s$ OR $t_0 \notin \mathbb{N}$ OR $t_0$ is a punctuation:
        RETURN An empty tree.
    ELSE:
        RETURN $T$.
ELSE:
    IF the root of $T$, denoted $r$, is such that $r \notin \mathbb{P}$:
        RETURN An empty tree.
    ELSE IF $r$ has only one child $c_0$:
        Apply Algorithm 2.2 to the subtree rooted at $c_0$.
        RETURN The pruned subtree rooted at $c_0$.
    ELSE:
        Prune all subtrees rooted at $r$ by applying Algorithm 2.2 recursively.
        Remove from $r$ the subtrees which have empty pruned trees.
        RETURN The resulting pruned tree, denoted $\hat{T}$.
```

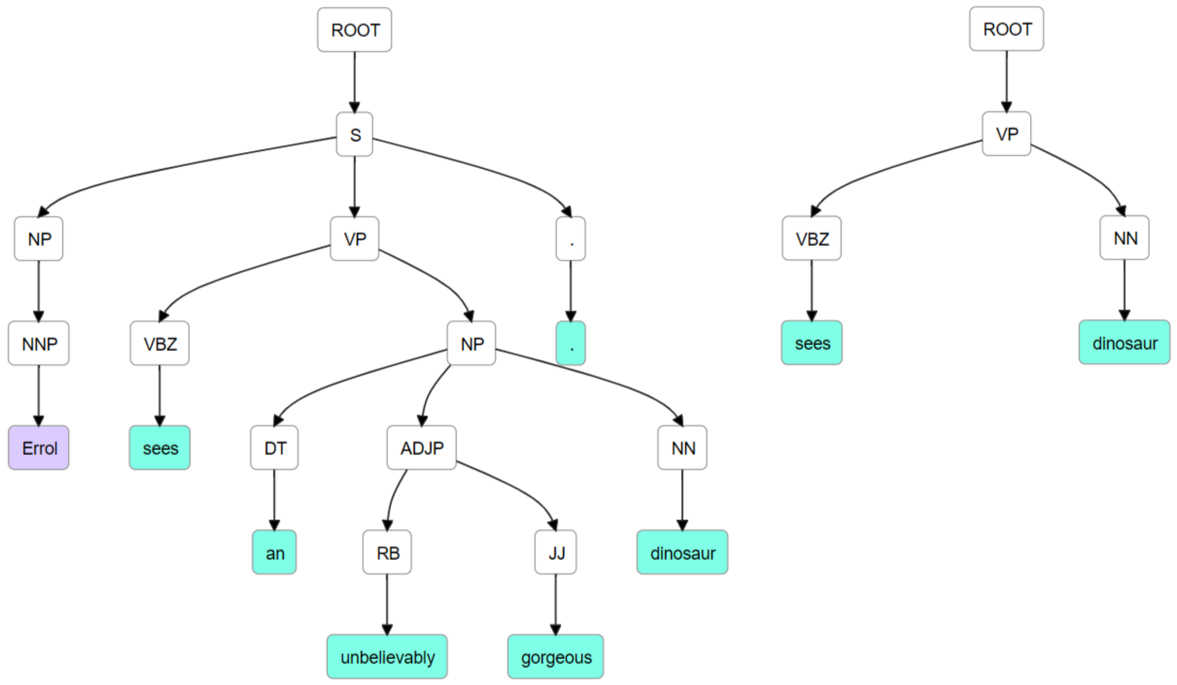


Figure 2.6: The *CBPT* of the sentence "*Errol sees an unbelievably gorgeous dinosaur.*" is shown on the left. The pruned *CBPT* is shown on the right. In this example, we remove *PERSON* name entities, *stopwords*, punctuations, and we are only interested in *NNs* and *VPs*. Consequently, "*Errol*" marked as *PERSON* is removed, "*an*" as in a *stopword* is removed, "*unbelievably gorgeous*" marked as *ADJP* is removed, and finally "." as in a punctuation is removed.

Stage 3: Semantically similar words across documents are identified in this stage. Computing lexical similarities is another modular design of *DSCTP*, and *DSCTP* is independent of lexical similarity methods. In our implementation of *DSCTP*, we have tried two different types of lexical similarity methods, *NASARI* [21] with cosine similarity, and *ADW* [114]. *NASARI*, as mentioned in Chapter 1, is a *Word2vec* lexical representation relying on neural network models, while *ADW* [114] works by utilizing *semantic networks*.

The threshold θ_w for lexical similarities is an input parameter for this stage. Determining this threshold is an empirical issue. From our experiments in Section 2.5, θ_w will need to vary according to the scenario, and also depends on the lexical similarity method in use. Since the main objective of developing *DSCTP* is to evaluate the effectiveness of *GPs* in representing document semantics rather than investigating into every technical detail of *DSCTP*, exploring more rigorous and effective approaches to determine this threshold is considered as one of our future works.

The running time of this stage is $O(NM)$, where N and M are the number of words in $\hat{T_{ik}}$ and $\hat{T_{jh}}$.

Stage 4: The assignment of weights for a *Syn-Lex* graph merits a discussion. There are two different types of edges in a *Syn-Lex* graph. One is induced from *CBPTs* encoding syntactic relations. The other is induced from lexical similarities. This stage can be thought of as preparation for **Stage 5** where semantically similar *GPs* are extracted. To facilitate this extraction, the weights on lexical similarity induced edges are set to a large-enough value, while the weights on syntactic relation induced edges are set to 1. This means that in **Stage 5**, by computing *MWBCs*, the path lengths of extracted *GPs* are minimized, and also it is guaranteed that in each *MWBC* there are exactly two *GPs*.

The running time of this stage is proportional to the number of identified semantically similar words in **Stage 3**.

Stage 5: *MWCBs* are computed by utilizing an algorithm provided in [71]. The time complexity of this algorithm is $O(m^2n + mn^2 \log n)$, where n is the number of vertices and m is the number of edges. The major issue in this stage is that a *MWCB* may not be unique for a graph, which implies that by finding only one *MWCB* some potential semantically similar *GPs* may have been missed. On the other hand, if, instead, we find all *MWCBs*, the union of all *MWCBs* can be exponential-sized [156]. Moreover, enumerating all basic cycles is required, and thus, the running time of this stage can be exponential. In our current design, we choose to trade off some potential semantically similar *GPs* and keep this step polynomial. Investigating trade-off solutions is a future work. The weight assignment in **Stage 4** guarantees that each resulting basic cycle will contain exactly two lexical-similarity-relation edges, which thereby guarantees that each basic cycle will contain at least one, and at most two, terminals in each sentence.

Stage 6: As discussed in Chapter 1, the path length of a *GP* reflects its significance. Then the significance of a pair of *GPs* in a 1-homology-class is naturally encoded in the two path lengths of the *GPs*. Without such significance, mere semantic similarities of *GPs* could not fairly reflect the document semantics. An example is shown in Figure 2.7.

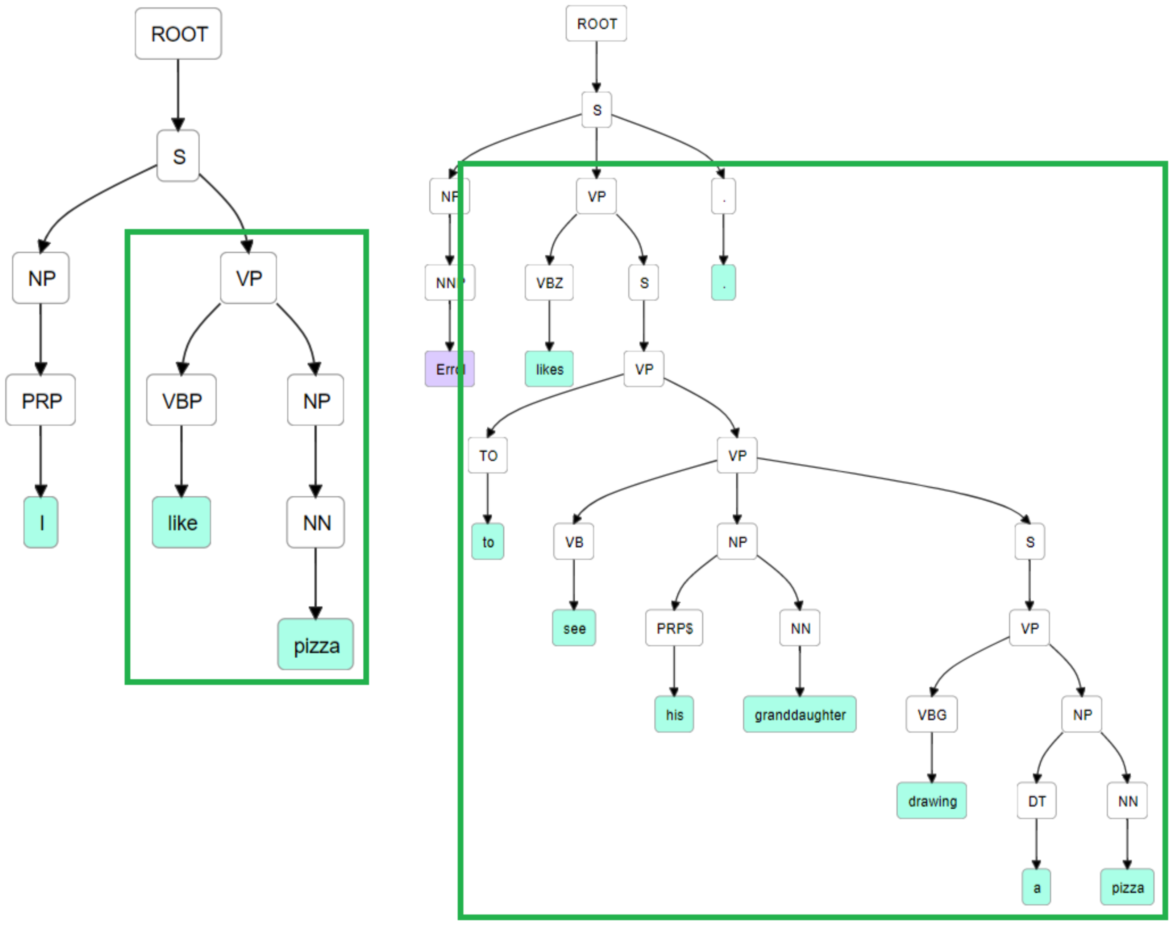


Figure 2.7: Both sentences contain "like" and "pizza", but due to the loose syntactic relation in the right sentence the two "*VP*"s in the two sentences respectively actually are not semantically similar. If the two "*like*"s and the two "*pizza*"s form a basic cycle, even though the two pairs of words are exactly the same, this cycle should not contribute significantly to the document semantic similarity.

Computing the significance weight is another modular design of *DSCTP*. As long as an approach can provide a reasonable value reflecting the significance of semantically similar *GP* pairs, then it can be used. Here we propose an empirical approach: let a, b

be the two sentences in two documents respectively; let c_1 be a basic cycle involving a, b; and let w_a, w_b be the two shortest path lengths of the two *GPs* contained in c_1; then the significance weight is computed by

$$w_{c1} = \frac{\theta_{s1}}{e^{w_a^{\theta_{s2}} + w_b^{\theta_{s2}}}} \tag{2.1}$$

where $\theta_{s1}, \theta_{s2} > 0$ are predetermined parameters for adjusting the range of w_{c1}. This formula is a variant of *harmonic mean* which gains a high value only when both w_a and w_b are low.

The running time of this stage depends on what method is used, and typically is proportional to the number of minimum basic cycles obtained from **Stage 5**.

Stage 7: Note that the *lifespan* of a 1-homology-class directly reflects the **semantic similarity** between the *GPs* contained in the 1-homology-class. When computing *lifespans*, *births* are determined by the filtration values, while *deaths* may not correspond to certain moments because, firstly, we do not consider higher dimensions; and secondly, death moments may not exist regardless of higher dimensional simplicies. As a convention, the *lifespans* are cut off by some point, and this cut-off point is determined by the range of lexical similarity. For example, if the lower bound of lexical similarities is 0, then the filtration values cannot exceed 1. Thus, 1 is a good point for the cut off.

The running time of this stage is proportional to the minimum basic cycles obtained from **Stage 5**.

Stage 8: We exponentiate the lifespans of the minimum basic cycles to emphasize the semantic similarities contributed by semantically similar *GPs*.

The running time of this stage is also proportional to the minimum basic cycles obtained from **Stage 5**.

Based on the running time of each stage, the running time of **Algorithm 2.1** is dominated by **Stage 5**, and can be exponential.

In the next section, the performance of *DSCTP* is evaluated.

2.5 Experimentation & Analysis

The performance of *DSCTP* is evaluated by comparison with human judges of the semantic similarity of documents. This is done by considering a set of documents, and for each pair of such documents, *DSCTP* will output a real value as its similarity score between those two documents. Likewise, human judges will provide a similarity score for each input document pair. Finally, the results produced by *DSCTP* will be compared with the results provided by the human judges.

We also compare *DSCTP* with a set of state-of-the-art word embedding methods, and most of them are based on neural network models. The methods in comparison are evaluated in the same way as *DSCTP*. The experimental results show that *DSCTP* is generally as effective as the word embedding methods, and in some cases, *DSCTP* significantly outperforms some of the state-of-the-art methods.

Next, we detail our experiments from four aspects: *datasets*, *methods in comparison*, *performance evaluation methods* and *experimental results*.

2.5.1 Datasets

Five datasets are used in our experiments:

Li30: *Li30* [89] consists of 30 sentence pairs, each of which is associated with a discerned rating by human judges reflecting the semantic similarity between the sentences. Note that *Li30* was built based on the celebrated *RG*65 dataset which consists of 65 word pairs each of which is associated with a human rating[16] of word similarity [123]. Thus, the original sentence pair set has 65 pairs. However, according to [99] and [25], a strong bias is exhibited in *RG*65. Specifically, two-thirds of the human ratings fall in the upper and lower quarters of the similarity range. Thus, the authors of [89] sampled 30 pairs out of the 65-pair set to make *Li30*. The human ratings of *Li30* range from 0 (minimum similarity) to 4 (maximum similarity).

[16] The human ratings range from 0 to 4, where 0 corresponds to the minimum word similarity and 4 corresponds to the maximum word similarity.

Lee1225 Human Ratings Statistics					
Mean	**Std**	**[1, 2)**	**[2, 3)**	**[3, 4)**	**[4, 5]**
1.63	0.70	968	173	66	18

Table 2.1: *Std* is short for the *standard deviation*. Each of the last four columns shows that number of document pairs whose scores fall into the corresponding range (as the title of the column). Both the *mean* and the counts of document pairs in the four ranges indicate a strong bias toward dissimilarity of documents.

Lee1225: *Lee1225* is another popular dataset specific to document semantics comparison. *Lee1225* is provided in [82] and consists of 50 documents selected from the Australian Broadcasting Corporation's News mail service. Each document pair (1225 document pairs in total) is associated with a human rating of document semantic similarity. The lengths of the documents vary from 51 to 126 words, and cover a number of broad topics. The human ratings of *Lee1225* range from 1 (minimum similarity) to 5 (maximum similarity).

Lee60: *Lee60* is a sampled subset of *Lee1225* that we have constructed. We observed that the human ratings of *Lee1225* have a strong bias. This observation is shown in Table 2.1 and Figure 2.8. To remove the bias as much as possible, we sample *Lee1225* in a similar approach as *Li30* in [89]. Specifically, we used a *systematic sampling* [150] over *Lee1225*. For the four ranges (i.e. **[1, 2)**, **[2, 3)**, **[3, 4)** and **[4, 5]**, we sample 15 documents from each of the ranges with equal intervals. Thus, in total 60 document pairs are collected for *Lee60*. The distribution of human ratings of *Lee60* is shown in Figure 2.9.

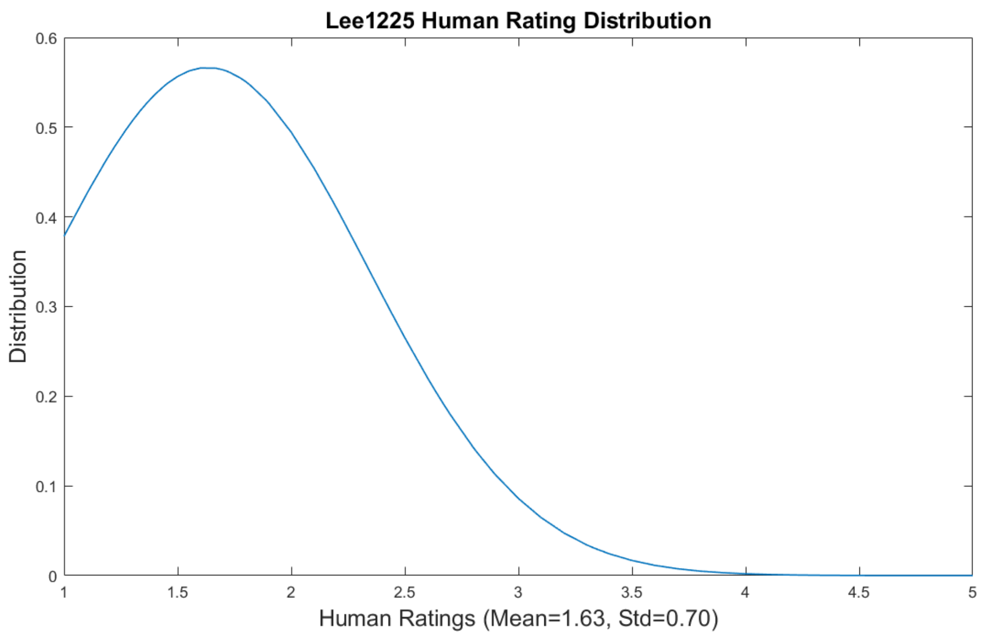


Figure 2.8: *Lee1225* human rating distribution. Mean is 1.63, and standard deviation is 0.70.

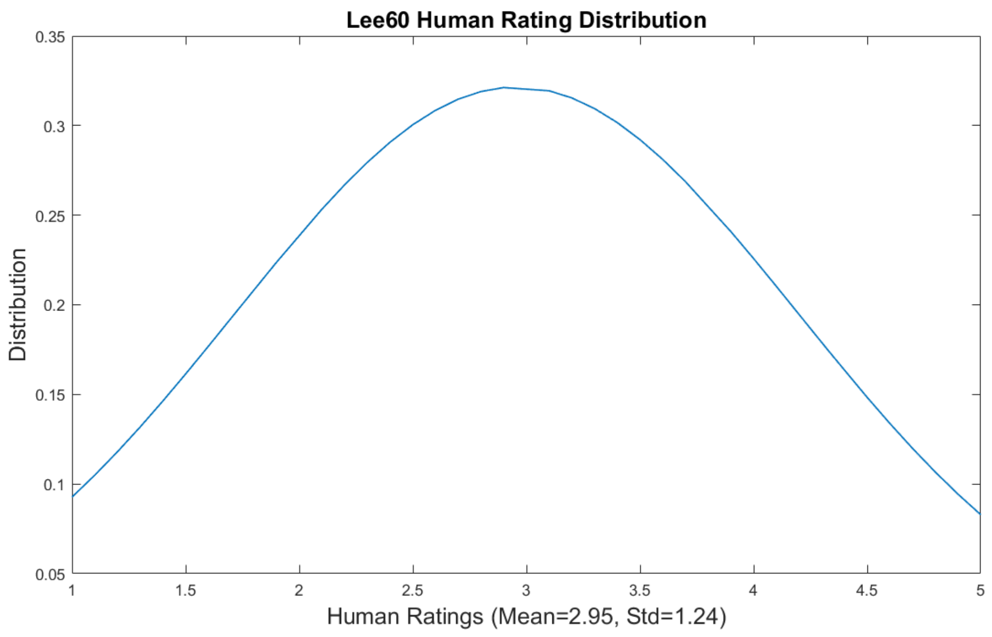


Figure 2.9: *Lee60* human rating distribution. Mean is 2.95, and standard deviation is 1.24.

STS2017: *STS2017* is provided by *SemEval-2017 Task1 Track5* [23] consisting of 250 sentence pairs with human ratings ranging from 0 (minimum similarity) to 5 (maximum similarity). Since *STS2017* has a basically balanced distribution, we do not further process the original data.

SICK *SICK* is provided by [94] consisting of 9840 sentence pairs with human ratings ranging from 1 (minimum similarity) to 5 (maximum similarity). Similar to *STS2017*, *SICK* has a balanced distribution, so we do not do any sampling.

2.5.2 Methods in Comparison

DSCTP is our document comparison method. As discussed earlier, *DSCTP* is of a modular design. Particularly, the lexical similarity method in *DSCTP* can have different choices. In our experiments, we test *DSCTP* with two different lexical similarity methods: one is *NASARI* [21] with cosine similarity, the other is *ADW* [114]. *DSCTP* is tested on 10 lexical similarity thresholds, from 0.1 to 1.0. The stop word list is provided by the *Gensim* [119]. The name entities for the parse tree pruning are: *LOCATION*, *ORGANIZATION* and *MISC*, which are defined by *CoNLL 2003 NER task* [118]. The *POS* tags for the parse tree pruning are listed as follows: {'S', 'SBAR', 'SBARQ', 'SINV', 'SQ', 'ADJP', 'ADVP', 'FRAG', 'RRC', 'PP','INTJ', 'NP', 'NP-TMP', 'NX', 'NAC', 'NN', 'NNS', 'NNP', 'NNPS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ', 'VP', 'JJ', 'JJR', 'JJS', 'RB', 'RBR', 'RBS', 'RP'}, where the tags are defined in the *Penn Treebank II Constituent Tags* [12].

As mentioned, *DSCTP* is compared with a set of state-of-the-art document semantics comparison methods (*DSCMs*). The state-of-the-art methods are all word-embedding-based *DSRs* discussed in Chapter 1 including *Word2Vec* [98], *NASARI* [21], *Doc2Vec* [80], *GloVe* [110], *fastText* [15], *LexVec* [126] and *Sent2Vec* [107]. Among these methods, *Word Mover's Distance (WMD)* [77] is equipped with *Word2Vec*, and all others work with *cosine similarity*.

2.5.3 Performance Evaluation Methods

Two evaluation methods are applied. One is *Spearman's correlation* [159] and the other is a classification task.

On *Li30*, *Lee60* and *STS2017*, we compute *Spearman's correlation* to evaluate the performance of the methods in comparison. Specifically, the *Spearman's correlations* are computed against human ratings. *Spearman's correlation* is computed by

$$\rho_{r_X, r_Y} = E[r_X \cdot r_Y] = \frac{cov(r_X, r_Y)}{\sigma_{r_X} \sigma_{r_Y}} \tag{2.2}$$

where X and Y denote two scorings on the same document comparison task (i.e. for each pair of documents, X and Y respectively provide a similarity score), r_X and r_Y denote the corresponding rankings of the two scorings X and Y (i.e. assigning a rank to each score), ρ_{r_X, r_Y} denotes the *Spearman's correlations* between X and Y, $cov(r_X, r_Y)$ denotes the *covariance* between r_X and r_Y, and $\sigma_{r_X}, \sigma_{r_Y}$ denote the *standard deviations* of r_X and r_Y.

On *Lee1225*, we apply a classification task as follows[17]. In this task the pairs of documents from the dataset will be partitioned into two groups. One group (denoted by $\mathbb{G}_s$) contains the document pairs which are determined to be semantically similar by human judges, and the other group (denoted by $\mathbb{G}_d$) contains the pairs determined to be dissimilar. Since in *Lee1225* human ratings are ranging from 1 to 5, where 1 indicates the minimum similarity and 5 indicates the maximum similarity, naturally it is reasonable to assume that the document pairs with human ratings falling into the range $[1, 2.5]$ are definite dissimilar, that the document pairs with human ratings

[17] Several studies [172] [164] [6] [122] have shown that *Spearman's correlation* can be significantly impacted by bias (e.g. *negativity bias* [69]). *Lee1225* is a strongly biased dataset as shown in Figure 2.8. Thus, we do not apply *Spearman's correlation* on *Lee1225*, and instead we use the classification task. Although the classification task may also be affected by bias because of the uneven numbers of document pairs in the two classes (i.e. positive and negative), it certainly shows the performance of document comparison methods with more details (e.g. *type I* and *type II* errors [134]) rather than a mere number, and also we do not have to sample the original dataset. Better statistical measures may exist; however, since statistical measurement is not a primary concern in this dissertation, we leave this topic to others.

falling into the range $[3.5, 5]$ are definite similar. The document pairs with human ratings between 2.5 and 3.5 have ambiguity, and are not in our consideration. Thereby, we obtain $\mathbb{G}_s$ and $\mathbb{G}_d$. For each group, a *DSCM* computes a similarity value for each document pair in this group. The confidence interval for a given *confidence* is computed based on these similarity values. In our experiments, the confidence is set to 99%. The confidence interval for $\mathbb{G}_s$ is denoted by $\mathbb{I}_s$, and $\mathbb{I}_d$ denotes for $\mathbb{G}_d$. Thereby, $\mathbb{I}_s$ and $\mathbb{I}_d$ are the two classes for the classification task. If the similarity value given by a *DSCM* for a pair of documents in $\mathbb{G}_s$ is less than or equal to the supremum of $\mathbb{I}_d$ (denoted by $\sup \mathbb{I}_d$), or the similarity value of a pair of documents in $\mathbb{G}_d$ is greater than or equal to the infimum of $\mathbb{I}_s$ (denoted by $\inf \mathbb{I}_s$), then this document pair is said to be *misclassified*. On the other hand, if the similarity value of a pair of documents in $\mathbb{G}_s$ is greater than or equal to $\inf \mathbb{I}_s$ or the similarity value of a pair of documents in $\mathbb{G}_d$ is less than or equal to $\sup \mathbb{I}_d$, then this document pair is said to be *correctly classified*. The categorization of misclassified pairs (into false positive or false negative) depends on which group, $\mathbb{G}_s$ or $\mathbb{G}_d$, is chosen as the positive class. The performance of a *DSCM* in the classification task is evaluated by error rates, recall, precision and F1 scores[18]. Specifically, the error rate will be computed for each class, and also the F1 scores will be computed for both the case with $\mathbb{I}_s$ chosen as the positive class and the case with $\mathbb{I}_d$ chosen as the positive class.

The objective of the classification task is to test if a document comparison method could distinguish between semantically similar document pairs and dissimilar pairs. It can be considered as a simpler task than correlation tasks (e.g. *Pearson's correlation*, *Spearman's correlation* and *Kendall's tau*). This task is introduced as a complement to *Spearman's correlation*. The correlation tasks require the evaluated methods to have monotonic relationships with human judges' results to gain high

[18] Recall is defined by recall $= \frac{tp}{tp+fn}$, where tp denotes the number of true positives and fn denotes the number of false negatives. Precision is defined by precision $= \frac{tp}{tp+fp}$, where fp denotes the number of false positives. F1 is defined by F1 $= 2\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$. Readers are referred to [132] for more details about recall, precision and F1.

scores, while the classification task does not strictly require such monotonic relationships to gain high scores but does require the distributions of documents similarity values for the two classes to be significantly distinguishable from each other. A critical motivation behind the classification task is that in human judges' results a pair of documents may have been given significantly different scores, sometimes even opposite. Thus, if there are no large-enough amount of human judges on each pair of documents, the average human judge scores for a pair of document may not fairly reflect the similarity relationship between the documents. If so, then the effectiveness of correlations is gravely undermined. Moreover, for human judges, it may be difficult to clearly distinguish between close quantitative similarity levels. For example, if one has to score a pair of documents with respect to their semantic similarity from 1 to 5, then how much is the difference between the scores 1 and 2 in this person's mind? Also, how much is the difference in the amount of similarity between the change from 1 to 2 and the change from 4 to 5? From the value point of view, the change from 1 to 2 has no difference from 4 to 5, then from the cognitive point of view, is it still true? These questions are left to *Linguistics* and *Cognitive Science*. What is emphasized here is that correlations may not be perfect in evaluating *DSCMs*, and the classification task can enhance the performance evaluation from a different perspective.

2.5.4 Experimental Results

To provide an intuitive understanding of the behaviors of *DSCTP* when comparing documents' semantics, we show some statistics of *DSCTP* on *Lee1225*. We pick *Lee1225* because it contains document pairs instead of sentence pairs. In Table 2.2, we show statistics of *Lee1225* including the average number of sentences per document and the average number of non-trivial words (i.e. non-stopword and non-punctuation words) per document. To demonstrate *DSCTP*'s behaviors, we use a typical configuration of *DSCTP*: *DSCTP* integrated with *NASARI* setting the lexical similarity threshold to 0.3. The distribution of the numbers of minimum basic cycles extracted

Lee1225 Statistics	
Avg. Num. of Sent. per Doc.	3
Avg. Num. of Non-trivial Words	45

Table 2.2: Statistics of Lee1225

from each document pair is shown in Figure 2.10. The distribution of pairwise document similarity values is shown in Figure 2.11. The distribution of the pairwise sentence similarity values which contribute to the document similarities is shown in Figure 2.12. From these three figures, it is straightforward to observe that these distributions are all tailed, which follows the *Zipf's law* [117].

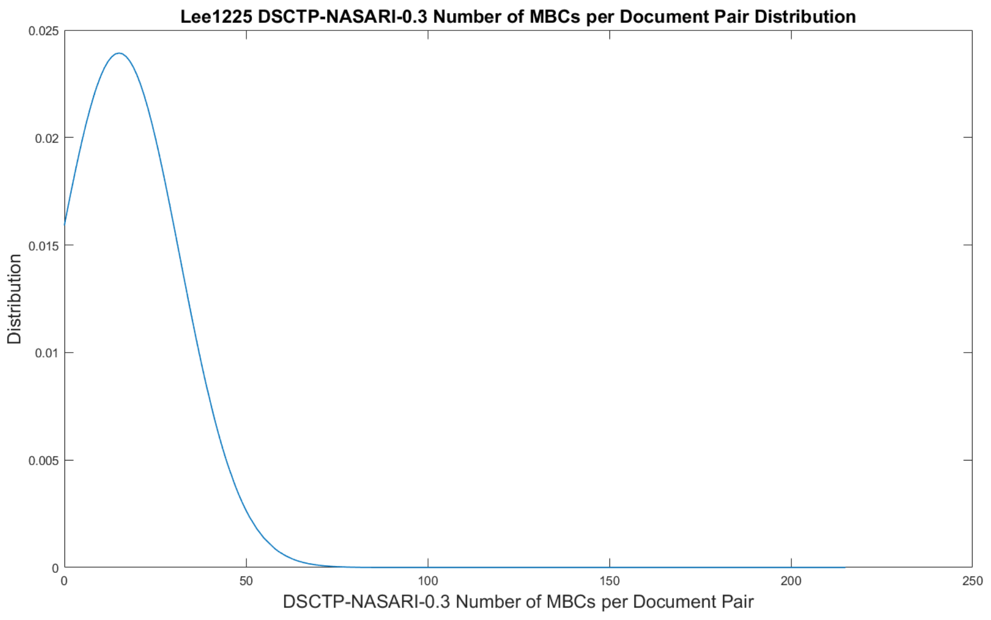


Figure 2.10: The distribution of the number of minimum basic cycles extracted from each document pair. The mean value is 15, and the standard deviation is 17.

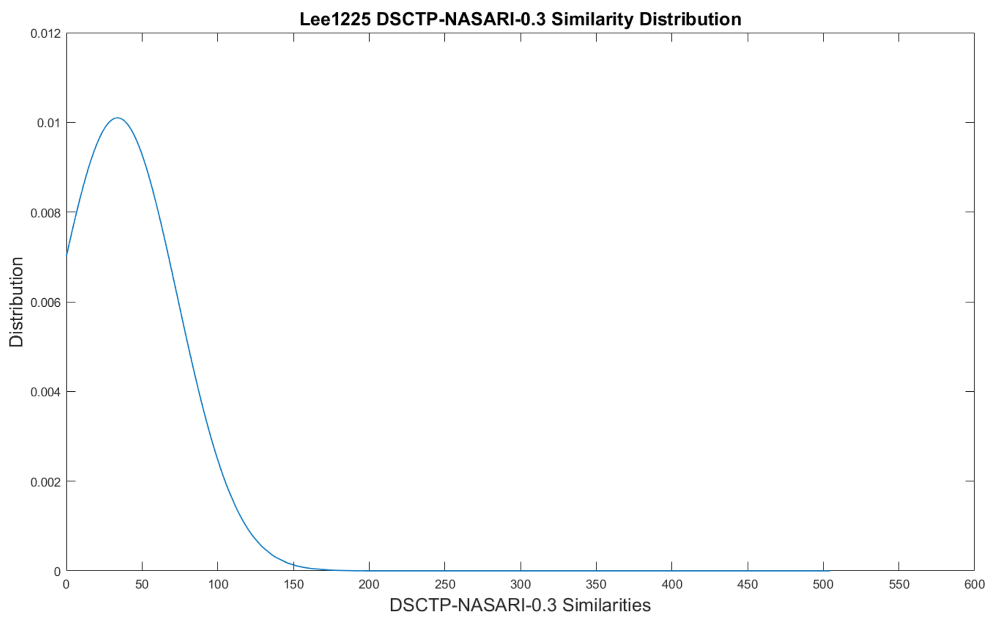


Figure 2.11: The distribution of pairwise document similarity values. The mean value is 33.76, and the standard deviation is 39.50.

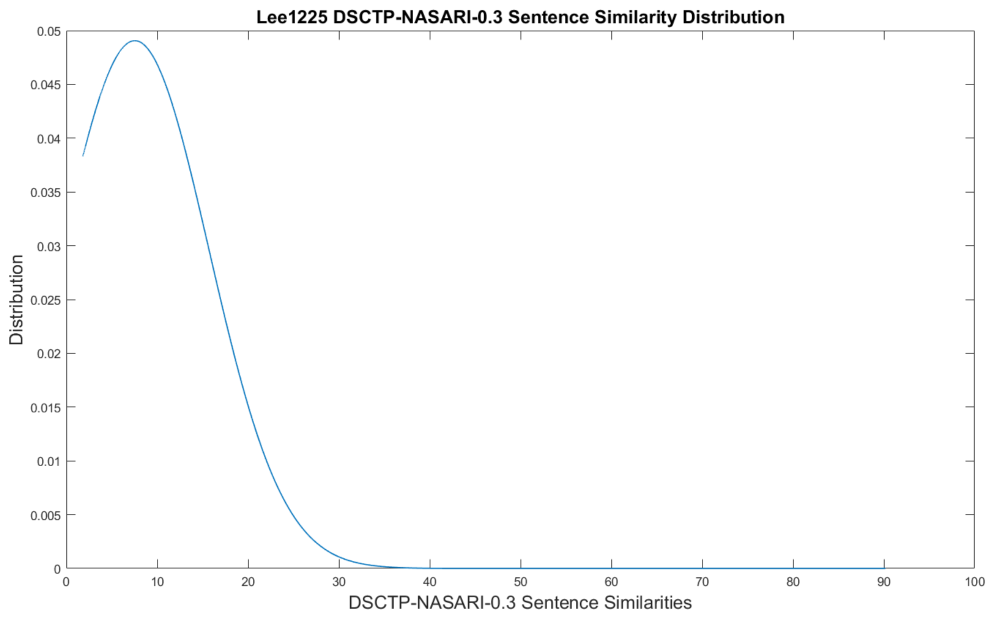


Figure 2.12: The distribution of the pairwise sentence similarity values. The mean value is 7.54, and the standard deviation is 8.13.

Before proceeding to the experimental results, we briefly consider the empirical running time of *DSCTP*. Our runtime hardware environment is as follows:

- Intel Core i7-8850H CPU @ 2.60GHz (6 physical cores with hyperthreading)
- 64GB DDR4 2400MHz Memory
- 1TB NVMe PCIe M.2 SSD

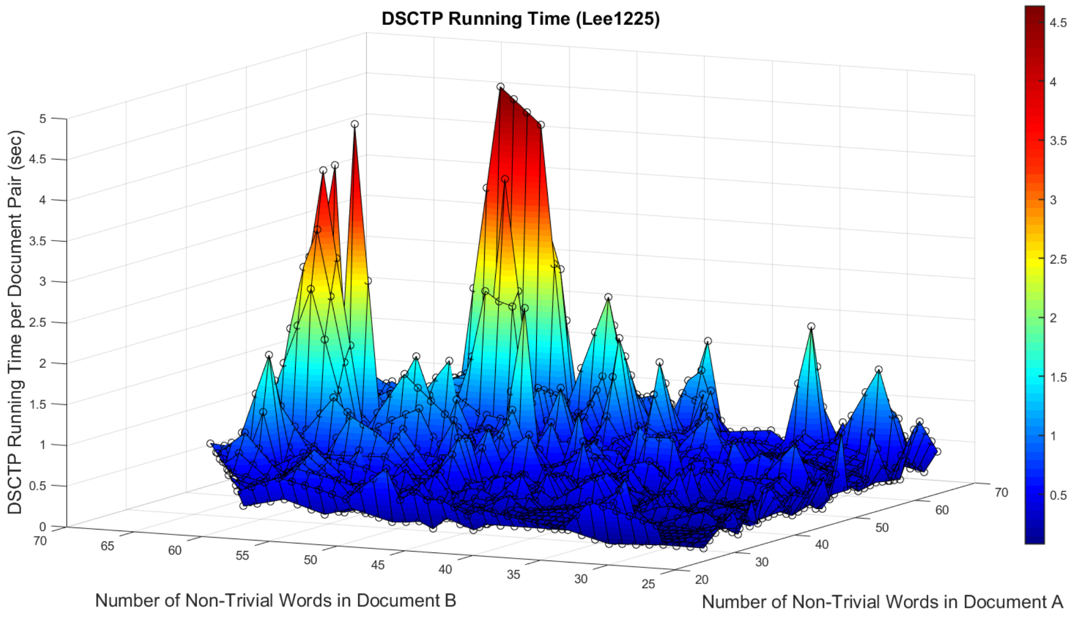


Figure 2.13: The relationship between the number of non-trivial words in two input documents and the running time of *DSCTP* computing the semantic similarity for the two documents on *Lee1225*.

We run *DSCTP* with *NASARI* setting the lexical similarity threshold to 0.3 on *Lee1225*. The running time statistics are illustrated in Figure 2.13. In this figure, the maximum running time is 6.14 seconds for comparing one pair of documents, the minimum running time is 0.08 seconds, and the average running time is 0.54 seconds.

The average running time of *Doc2Vec* on *Lee1225* comparing one pair of documents is 0.09 seconds. The average time of *NASARI* is 0.07 seconds. The average running time of *Sent2Vec* is 0.006 seconds. The average running time of *LexVec* is 0.003

seconds. The average running time of *fastText* is 0.003 seconds. The average running time of *GloVe* is 0.014 seconds. These running times are based on pre-trained models. The training times are not counted into the running times; however, the training times can be considerable. Our *DSCTP*, on the other hand, does not require training. In addition, pre-trained models can be large (e.g. the pre-trained *NASARI* model is 7GB). Loading such models can also take a considerable time. Such model loading times are not counted into the running times.

Next, we detail our experimental results.

The experimental results are summarized in Tables 2.3 and 2.4. Specifically, as mentioned earlier, we use a classification task to test the considered methods on *Lee1225*. Table 2.3 summarizes the results for those experiments. For other datasets, *Lee60*, *Li30*, *STS2017* and *SICK*, we compute a Spearman correlation for each method, and summarize these results in Table 2.4. Note that the notation, for example *DSCTP-N-0.4*, in both tables denotes our *DSCTP* method integrated with *NASARI* setting the lexical similarity threshold to 0.4, and similarly, for example, *DSCTP-A-0.3* denotes *DSCTP* with *ADW* setting the threshold to 0.3. We discuss some of the results below.

In Table 2.3, we summarize the error rate, precisions (denoted by **P**), recalls *denoted by* ***R*** and *F1* scores for each method. For each trial, we respectively consider $\mathbb{G}_d$ and $\mathbb{G}_s$ as the positive class to compute precisions, recalls and *F1* scores.

From the results shown in Table 2.3, the best performance of *DSCTP* is better than, and sometimes similar to, the state-of-the-art methods. Specifically, *DSCTP* with *NASARI* at 0.2 threshold reaches the lowest possible error rate for $\mathbb{G}_s$, *DSCTP* with *ADW* at 0.4 and 0.5 thresholds provide nearly the lowest error rates for $\mathbb{G}_s$. Similarly, for $\mathbb{G}_d$, *DSCTP* with *NASARI* at 0.4 and 0.5 provides nearly the lowest error rates though *NASARI* gives the best. Considering both $\mathbb{G}_d$ and $\mathbb{G}_s$, *DSCTP* with *NASARI* at 0.2, 0.3, 0.4, and *DSCTP* with *ADW* at 0.4 and 0.5 provides performance as good as the best performance given by the state-of-the-art methods such as *NASARI*, *GloVe* and *Sent2Vec*, and outperform other state-of-the-art methods. With respect to precisions, recalls and *F1* scores, *NASARI* gives the best results, though *DSCTP*

Lee1225								
	Error Rate		**$\mathbb{G}_d$ as positive**			**$\mathbb{G}_s$ as positive**		
Methods	$\mathbb{G}_s$	$\mathbb{G}_d$	**P**	**R**	**F1**	**P**	**R**	**F1**
DSCTP-N-0.1	0.1078	0.0870	0.85	0.99	0.91	0.87	0.19	0.30
DSCTP-N-0.2	**0.0000**	0.0420	0.94	**1.00**	0.97	**1.00**	0.36	0.53
DSCTP-N-0.3	0.0217	0.0110	0.98	**1.00**	0.99	0.97	0.70	0.81
DSCTP-N-0.4	0.0434	0.0082	**0.99**	**1.00**	0.99	0.93	0.75	0.83
DSCTP-N-0.5	0.0869	0.0091	**0.99**	**1.00**	0.99	0.88	0.74	0.80
DSCTP-N-0.6	0.1304	0.0210	0.97	0.99	0.98	0.81	0.53	0.64
DSCTP-N-0.7	0.2391	0.0301	0.97	0.99	0.98	0.71	0.45	0.55
DSCTP-N-0.8	0.3261	0.0228	0.98	0.99	0.98	0.63	0.50	0.56
DSCTP-N-0.9	0.3261	0.0283	0.97	0.99	0.98	0.63	0.45	0.52
DSCTP-N-1.0	0.3261	0.0274	0.97	0.99	0.98	0.63	0.45	0.53
DSCTP-A-0.1	0.5652	0.5836	0.41	0.95	0.58	0.43	0.03	0.06
DSCTP-A-0.2	0.5652	0.5662	0.43	0.95	0.60	0.43	0.03	0.06
DSCTP-A-0.3	0.1522	0.0164	0.98	0.99	0.98	0.77	0.57	0.66
DSCTP-A-0.4	0.0046	0.0217	**0.99**	**1.00**	**1.00**	0.96	0.84	0.89
DSCTP-A-0.5	0.0064	0.0217	**0.99**	**1.00**	**1.00**	0.96	0.78	0.86
DSCTP-A-0.6	0.1739	0.0119	**0.99**	0.99	0.99	0.76	0.66	0.70
DSCTP-A-0.7	0.2391	0.0530	0.93	0.99	0.96	0.71	0.32	0.44
DSCTP-A-0.8	0.2391	0.0511	0.93	0.99	0.96	0.71	0.33	0.45
DSCTP-A-0.9	0.2391	0.0438	0.94	0.99	0.96	0.71	0.36	0.48
DSCTP-A-1.0	0.2391	0.0384	0.95	0.99	0.97	0.70	0.38	0.50
Doc2Vec	0.1434	0.1304	0.79	0.99	0.88	0.81	0.14	0.24
NASARI	0.0037	**0.0000**	**0.99**	**1.00**	**1.00**	**1.00**	**0.89**	**0.94**
GloVe	**0.0000**	0.0420	0.92	**1.00**	0.96	**1.00**	0.37	0.54
WMD	0.0652	0.0292	0.94	0.99	0.97	0.91	0.48	0.63
LexVec	0.0217	0.0384	0.93	**1.00**	0.96	0.97	0.42	0.58
fastText	0.0435	0.1059	0.82	**1.00**	0.90	0.94	0.21	0.34
Sent2Vec	**0.0000**	0.0173	0.97	**1.00**	0.98	**1.00**	0.58	0.73

Table 2.3: The classification task results on *Lee1225*. The best score for each category is bold.

(with *ADW* at 0.4 threshold, and with *NASARI* at 0.4) provides similar performance, and outperforms other state-of-the-art methods. From an introspective perspective, *DSCTP* with *NASARI* at 0.3, 0.4, 0.5 and 0.6 perform similarly, and *DSCTP* with *ADW* at 0.3, 0.4, 0.5, and 0.6 perform similarly.

Based on these results, we particularly highlight two observations:

	Spearman Correlation			
Methods	Lee60	Li30	STS2017	SICK
DSCTP-N-0.1	0.56	0.58	0.10	0.22
DSCTP-N-0.2	0.64	0.70	0.31	0.37
DSCTP-N-0.3	0.73	0.73	0.45	0.44
DSCTP-N-0.4	0.79	0.82	0.56	0.47
DSCTP-N-0.5	0.82	**0.86**	0.63	0.48
DSCTP-N-0.6	**0.85**	0.80	0.66	0.49
DSCTP-N-0.7	0.82	0.71	0.68	0.49
DSCTP-N-0.8	0.77	0.68	0.70	0.50
DSCTP-N-0.9	0.77	0.66	0.70	0.50
DSCTP-N-1.0	0.77	0.66	0.70	0.50
DSCTP-A-0.1	0.09	0.12	0.04	0.11
DSCTP-A-0.2	0.11	0.17	0.03	0.12
DSCTP-A-0.3	0.35	0.40	0.20	0.31
DSCTP-A-0.4	0.46	0.73	0.40	0.44
DSCTP-A-0.5	0.61	0.75	0.54	0.49
DSCTP-A-0.6	0.68	0.77	0.60	0.50
DSCTP-A-0.7	0.67	0.75	0.64	0.51
DSCTP-A-0.8	0.71	0.72	0.67	0.50
DSCTP-A-0.9	0.72	0.73	0.67	0.52
DSCTP-A-1.0	0.73	0.73	0.67	0.52
Doc2Vec	0.57	0.78	0.75	0.56
NASARI	0.79	0.83	0.71	0.55
GloVe	0.81	0.67	0.72	0.54
WMD	0.82	0.78	**0.80**	0.57
LexVec	0.77	0.72	0.73	**0.61**
fastText	0.71	0.79	0.72	0.54
Sent2Vec	0.83	0.82	0.74	0.55

Table 2.4: Spearman correlations for *Lee60*, *Li30*, *STS2017* and *SICK*. The best score for each category is bold.

#1 *DSCTP* performs well with both *NASARI* and *ADW*.

#2 The performance of *DSCTP*, regardless of the choice of lexical similarity method, is not sensitive to the settings of the lexical similarity threshold.

In Table 2.4, for *Lee60* and *Li30*, the best scores are both provided by *DSCTP*. The former is given by *DSCTP* with *NASARI* at 0.6, and the latter is given by *DSCTP* with *NASARI* at 0.5. The best scores provided by *DSCTP* with *ADW* on these two datasets are close to the state-of-the-art methods, and outperform some of them such as *Doc2Vec* and *fastText* on *Lee60*, and *GloVe* and *LexVec* on *Li65*. On *STS2017* and *SICK*, the best performance provided by *DSCTP* is close to the state-of-the-art methods. From the results on these four datasets, the observations #1 and #2 from above also hold.

In summary, we conclude that, first, *DSCTP* can outperform, or perform similarly to, the stat-of-the-art methods; second, *DSCTP* can perform well with different lexical similarity methods, and its performance is independent of the choice of lexical similarity method; and third, regardless of the choice of lexical similarity method, *DSCTP* is not sensitive to the settings of the lexical similarity threshold.

We note that the third conclusion unveils a significant property of *DSCTP* when it is applied to real scenarios. As discussed earlier, the threshold for lexical similarities is a critical input parameter of *DSCTP*, and determining this threshold in a rigorous and effective way deserves future work. Nonetheless, with this significant property the utility of *DSCTP* will not rely on subtle parameter tunings.

Although *DSCTP* shows all positive results in our experiments, when examining the details of the experimental results, we found a couple of particular cases in which *DSCTP* may not work so well.

First, consider the following two sentences (both from *STS2017*):

- "*There are dogs in the forest.*"
- "*The dogs are alone in the forest.*"

The human rating for these two sentence is 4 which indicates that they are very similar in semantics. However, *DSCTP* gives a relatively low similarity score on this sentence pair because there is only one pair of *GPs* that are semantically similar (i.e. "*dogs ... forest*"). Even though the two *GPs* are exactly the same in the two sentences, by using *DSCTP*, the document semantic similarity is only computed based on this pair of *GPs*. The underlying reason is that the lengths of the sentences are very short, which leads to a deficiency of semantically similar *GP* pairs.

Second, consider another pair of sentences (both from *STS2017*):

- "*A man is carrying a canoe with a dog.*"
- "*A dog is carrying a man in a canoe.*"

In this pair of sentences, *DSCTP* will extract three pairs of semantically similar *GPs*, and thus *DSCTP* will give a high similarity score to this pair of sentences. However, human judges give 1.8 which is not a high similarity score. From a human understanding perspective, these two sentences have different meanings because they have distinct *argument structures* (i.e. dependency relations between words). The subjects and the objects of "*carrying*" in the two sentences are opposite. Such a difference is not captured by the constituency structures.

To address the first issue, sentence lengths may have to be taken into consideration; and to address the second issue, we may have to also consider dependency relations while using constituency relations. We leave these to future works.

2.6 Conclusion

DSCTP as a new document semantics comparison method has been shown from the experiments to be highly effective. Thereby, the effectiveness of *GPs* in reflecting document semantic similarities is justified, which further implies that *GPs* will be effective in reflecting document semantics. That paves the way toward single document semantic representations by utilizing *GPs*, and it is addressed in Chapter 3 and 4.

Chapter 3
ABSTRACT PHRASE VECTORS

3.1 Introduction

The problem considered in this chapter is to construct a computational representation for the semantics of *a single document*. Converting input documents into computational representations before further processing is required by, or makes easier, solving many key problems in *NLP* and *IR* (e.g. *document indexing* [38], *document classification* [133] and *document clustering* [132]). From the discussions on *DSCTP* in Chapter 2, *generalized phrases (GPs)* have been shown to be effective to carry semantics in comparing two documents' semantics. However, *DSCTP* does not provide an explicit semantic representation for a single document. In this chapter, we introduce a new document semantic representation for a single document based on *GPs*, named **abstract phrase vector (APV)**.

The motivation behind *APVs* is straightforward. Since *GPs* in a document carry fragments of the semantics of this document, and a collection of such *GPs* can distinguish this document from other documents, it suggests utilizing a collection of *GPs* to represent the document semantics. In other words, the semantic representation of a document can be based on the *GPs* extracted from the document. Naturally, the quality of this representation depends in large measure on which *GPs* are extracted. Thus to develop an effective representation, determining which *GPs* should be extracted is a critical problem to be solved.

3.1.1 Use All *GPs*?

Theoretically, given a document, in each of its sentences, any two words and the shortest path (in the corresponding *CBPT*) between them forms a valid *GP* (Section

1.2). Then by enumeration, it is straightforward to extract all *GPs* from a document. We call these *GPs* a **complete collection** of *GPs* for the document. Thus, one possibility for the *GP*-extraction is to directly extract the complete collection of *GPs* from the document. The advantage of this complete *GP*-extraction approach is that nothing is required beyond parse trees. However, this approach has some intrinsic drawbacks.

First, in this approach some extracted *GPs* may not be meaningful in any sense. For example, *"play games"* is a typical *GP* that expresses a common-sense meaning, while *"time . . . fear"* could hardly be as meaningful as the former one.[1] More specifically, it would be difficult to find other *GPs* that have common-sense meanings and are semantically similar to *"time . . . fear"* (i.e. the chance that *"time . . . fear"* forms a meaningful constituent is not great).

Second, some *GPs* are more representative than others under a specific topic, which makes them more significant to reflecting document semantics. For example, consider the *GP*: *"computation . . . algorithms"*, which is a commonly seen *GP* under the topic *"computer science"*. Then, naturally in many documents under the topic *"computer science"*, there exist *GPs* semantically similar to *"computation . . . algorithms"* while under other topics (e.g. "*food*" and "*politics*") the *GP* *"computation . . . algorithms"* will rarely occur. Thus, *"computation . . . algorithms"* is representative to the topic *"computer science"*. When extracting *GPs* from documents, of course such *GPs* should be captured to reflect document semantics, for example, by the times of being extracted (i.e. the multiplicities of extraction). However, by means of extracting a complete collection of *GPs*, such representative *GPs* cannot be effectively captured as every *GP* in a document will be extracted exactly once[2]. This issue is severe especially when a

[1] These two *GPs* come from the following two sentences in the *BBC* dataset [53], documents *tech/238* and *tech/317* respectively, "The handheld console can **play games**, music and movies and goes on sale in Europe and North America next year." and "Sony has said it wanted to launch the PSP in Europe at roughly the same **time** as the US, but gamers will now **fear** that the launch has been put back."

[2] It happens that multiple *GPs* which have the same words, and even same length of paths, occur in

document is short (e.g. only a couple of sentences).

Third, similar to the second issue above, representative co-occurring *GPs* will also not be captured. Some *GPs* may not be individually representative to a topic, for example, "*string . . . tuning*". This *GP* occurs frequently in documents under both the topic "*instruments*" and the topic "*physics*", which makes it not representative to either of the two topics. On the other hand, if "*string . . . tuning*" co-occurs with another *GP* in a document, for example, "*standard . . . guitar*", then it will be more affirmative to say that this document has the topic "*instruments*", because the co-occurrence of "*string . . . tuning*" and "*standard . . . guitar*" is more frequent in the documents under the topic "*instruments*" than other *GP* co-occurrences, and such a frequent co-occurrence is specific to the topic "*instruments*" (i.e. this co-occurrence is representative to the topic "*instruments*"). However, by using a complete collection of *GPs*, no co-occurrence of *GPs* would be captured, because each *GP* in a sentence co-occurs with another *GP* in the same sentence exactly once.

Because of these above drawbacks, we conclude that a *GP*-extraction approach using a complete collection of *GPs* is not a good option.

3.1.2 A Better GP-Extraction

To seek a better *GP*-extraction approach, we propose three design guidelines:

A1. Each extracted *GP* should have a common-sense meaning.

A2. The *GP*-extraction should reflect the representativeness of individual *GPs* under a topic.

A3. The *GP*-extraction should reflect the representativeness of co-occurring *GPs* under a topic.

a document; however, what we are arguing is that for each *GP* it will not be particularly captured more than others by using a complete collection of *GPs*.

To satisfy these three principles, we design a "*GP* filtering" mechanism to extract a subset of *GPs* from the input document. Since the representativeness of individual and co-occurring *GPs* for the input document cannot be captured without prior information (e.g. relationships between topics and *GPs*), it is necessary to introduce an external set of documents (called the **background documents**). Specifically, for a given document, a set of background documents which are semantically similar to this document will be introduced as "*GP* filters". The filtering mechanism will work as follows: The input document will be compared with each background document to identify all semantically similar *GPs* across the two documents, and the *GPs* from those semantically similar *GPs* that belong to the input document will be extracted from the input document. In Chapter 2, it was discussed that the *GP* similarities across two documents can effectively reflect the document similarity. The stronger the *GP* similarities, the stronger the document similarity. We explain below why this method meets the three design guidelines:

For A1. Each extracted *GP* has at least one corresponding similar *GP* in a background document.

For A2. It is very likely that the given document A shares a topic with each similar document. Also, if a *GP* in A is representative under the topic, then the occurrences of this *GP* and its similar *GPs* in those similar documents, with a great probability, will be more than other *GPs*. Then, such a representative *GP* will be extracted more times than others, and its multiplicity reflects its representativeness under the topic.

For A3. If a document A has a topic, and the co-occurrence of some of its *GPs* (e.g. GP_{A1} and GP_{A2}) is representative under the topic, then the occurrences of such co-occurring *GPs* in A's similar background documents, with a great probability, will be more than other *GPs*. Thus, such representative co-occurring *GPs*, GP_{A1} and GP_{A2}, will be captured by their extraction multiplicities.

Clearly the background documents are a critical ingredient of *GP*-filtering. Different choices of the background documents may significantly affect which *GPs* are extracted. An ideal universal background document set would contain all meaningful *GPs* covering all possible topics. Finding such an ideal universal background document set will involve impossibly many documents, which leads to an unachievable task.

However, for an input document A, we provide a set of principles for selecting background documents: first, background documents should be of non-trivial lengths; second, the background document set should contain only *topics* of interest (e.g. in medical applications, topics highly related to medicine should be contained while topics in politics should not); third, the subsets of background documents for each topic should have equal size; and fourth, the size of the background document set should be sufficiently large. In the remainder of this dissertation, we assume that a background document set exists, and leave further consideration of the topic to future work.

3.1.3 GP Extraction

To realize the *GP*-filtering, a method that can find semantically similar documents, and extract *GPs* that contribute to the document similarities is required. The general approach of our *GP*-extraction method is illustrated in Figure 3.1. The green box shows an input document and its *complete collection* of *GPs*. The black box is the background document set in which multiple topics (e.g. *Food*, *Sports* and *Auto*) are covered. No doubt a *GP* can occur in multiple topics (e.g. *drink . . . beer* occurs in both *Food* and *Sports*). The arrows demonstrate the *GP*-filtering. By apply ing*DSCTP*, if a *GP* of the input document can be found to have a semantically similar counterpart in the background document set, then this *GP* will be extracted from the input document. The red box shows the resulting extracted *GPs* for the input document.

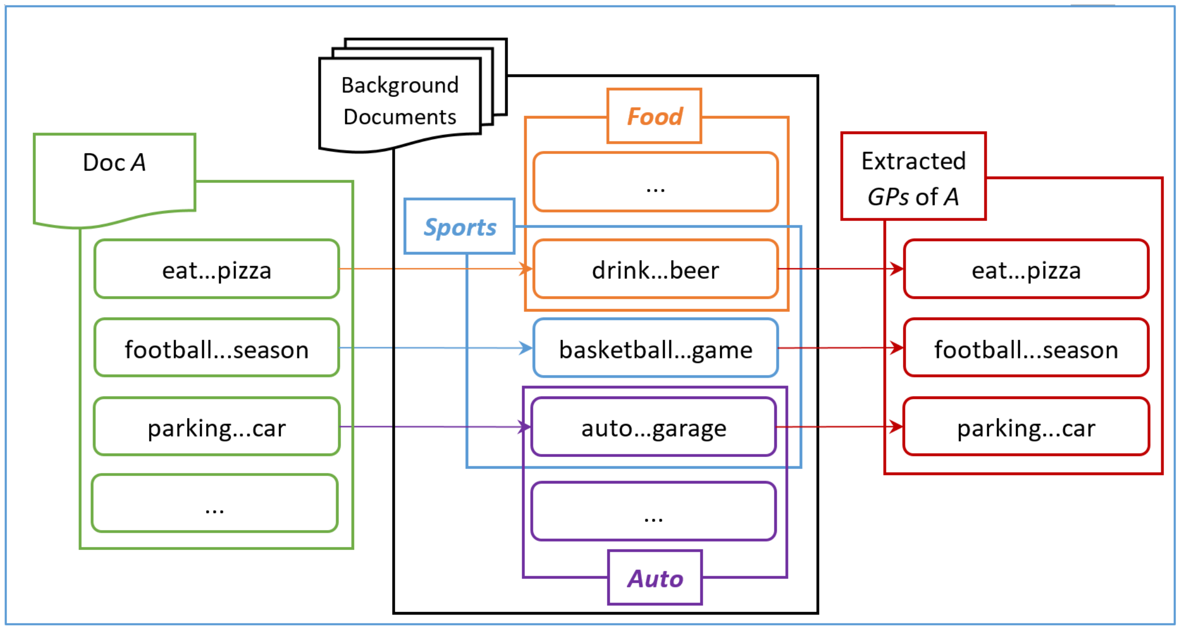


Figure 3.1: The *DSCTP*-based *GP*-extraction method.

Naturally, *DSCTP* is an ideal choice to accomplish that *GP*-filtering. Recall that *DSCTP*, to compute the semantic similarity between two documents, computes minimum basic cycles (as intermediate results), each of which captures two semantically similar *GPs* (Section 2.4). Straightforwardly, *GPs* can be extracted from those minimum basic cycles. Thus, a *DSCTP*-based *GP*-extraction method is utilized, named **GP-DSCTP**.

3.1.4 APV Construction

Given a collection of *GPs* extracted from an input document, these *GPs* are used to construct a semantic representation for the document. To make this representation convenient to use, we design it as a real-valued finite vector. This vector is named *abstract phrase vector (APV)*, and will meet the following two guidelines.

B1. Each *dimension* [3] of this vector should represent a distinct semantics of *GPs* (i.e. semantically similar *GPs* are always corresponding to the same dimension,

[3] Here the term "*dimension*" means a group of *GPs* that are semantically similar to each other. Note that a dimension is not a topic. In an *APV*, each such dimension corresponds to a real value.

and dissimilar *GPs* are always corresponding to different dimensions).

B2. The vector value for each dimension should reflect the significance of this dimension to the semantics of the input document.

To satisfy these requirements, we first extract a collection of *GPs* from the background documents, and then group the extracted *GPs* into clusters. The *GPs* in each cluster will be semantically similar to each other, and each cluster carries a distinct semantics. Thereby, each *GP* cluster forms a dimension for the APV, which thus satisfies **B1**. For each *GP* extracted from an input document, it will be classified into one or multiple *GP* clusters. For each such cluster, a real value is computed based on both the semantic similarity between the *GP* and the cluster, and on the path lengths of the *GPs* (recall that a *GP* consists of at most two words and the shortest path between them). This value reflects the significance of this cluster to the document semantics, thereby meeting **B2**. The greater the value, the more significant the cluster. Zero means that there is no significance, which can only occur when a document has no *GP* semantically similar to the *GPs* in a *GP* cluster. Finally, these values form the APV for the input document. We name this *GP*-clustering-based APV construction method **APV-GPC**.

3.1.5 Uniform APVs

To make $APVs$ convenient to use (e.g. to compute cosine similarities between documents), for every input document, the dimensions should be the same. Since the *GP* clustering is independent of any input document, then all input documents can use the same *GP* clusters (we assume that the background documents are well-selected). In other words, the dimensions are the same for every input document under consideration.

By utilizing our *DSCTP*-based *GP*-extraction method *GP-DSCTP* and *GP*-clustering-based APV-construction method APV-*GPC*, any given document will be given a real-valued and vector-formed semantic representation. The effectiveness of $APVs$ is discussed in Chapter 5. The experimental results there show that APV

provides competitive performance in *document clustering* compared to state-of-the-art *DSRs*.

The remainder of this chapter explains in detail *GP-DSCTP* and *APV-GPC*. Specifically, some fundamental definitions and concepts are provided in Section 3.2; the *GP*-extraction method is explained in Section 3.3; and the construction of *APVs* is explained in Section 3.4.

3.2 Definitions & Concepts

In this section, the formal definitions of *APV* and related fundamental concepts are provided.

Def: Abstract Phrase Vector (APV):

An **abstract phrase vector (APV)** for a document is a non-negative real-valued vector representing its semantics. Each dimension of this vector corresponds to a set of semantically similar *GPs*. Each such set of *GPs* is called an **abstract phrase (AP)**. The value corresponding to each *AP* is determined by the significance of the *AP* to the document semantics.

A set of *APVs* is said to be **uniform** if every *APV* has the same set of *APs* (i.e. every *APV* has the same dimensions).

Here, two *GPs* are semantically similar if their corresponding words are semantically similar according to the definition of *GP* in Section 1.2. This is regardless of the order of the words. Also, in an *APV*, each *AP*'s semantics are determined by its *GPs*, and the significance of the *AP* reflects the "proportion" of the document semantics that are covered by this *AP* (the significance will be further explained in Section 3.4.3).

3.3 Generalized Phrase Extraction

Extracting a collection of *GPs* from a given document is the first major step to construct an *APV* for the document. As discussed earlier, *DSCTP* is utilized in the

GP filtering, and thus the *GP* extraction will meet the design guidelines **A1**, **A2** and **A3**. Generally, this *DSCTP*-based *GP*-extraction approach works as follows: Given a document A and a background document set $\mathbb{B}$, *DSCTP* is applied to compare A and each background document in $\mathbb{B}$. Then, for each document comparison, a collection of *GPs* in A can be extracted from the *minimum basic cycles*.

The *GP-DSCTP* method is detailed in **Algorithm 3.1**.

Algorithm 3.1:

Given:

- A document A
- A set $\mathbb{B}$ of background documents

Seek:

A collection of *GPs* in A, each of which is semantically similar to at least one *GP* in the background documents.

```
LET A collection $\mathbb{GP}_A = \emptyset$.
FOR each document $b_i \in \mathbb{B} \setminus \{A\}$:
    Apply DSCTP to $A$ and $b_i$; From each minimum basic cycle produced by DSCTP, extract the GP that belongs to $A$ (a minimum basic cycle links two semantically similar GPs), denoted by $GP_i^A = (\{w_{i1}^A, w_{i2}^A\}, PathLen_{GP_i^A})$, where $\{w_{i1}^A, w_{i2}^A\}$ denotes the words, and $PathLen_{GP_i^A}$ denotes the path length between the words;
    Add $GP_i^A$ to $\mathbb{GP}_A$;
RETURN $\mathbb{GP}_A$.
```

Note that if the input document is also in the background document set, then that input document as a background document should be skipped when applying the *GP*-filtering method. Note also that $\mathbb{GP}_A$ is a collection rather than a set. The multiplicity of each *GP* can be retrieved from $\mathbb{GP}_A$. As discussed in Section 3.1, the multiplicities of *GPs* are critical for satisfying the requirements **A2** and **A3**.

In this algorithm, extracting *GPs* of A from *minimum basic cycles* is straightforward. As explained in Section 2.4, each minimum basic cycle contains two *GPs* from each of the two documents being compared. Then, by identifying the *GP* belonging to the input document, the *GP* can be extracted. An example of a *GP* to be extracted in a *minimum basic cycle* is shown in Figure 3.2. If a *GP* has only one word, then we expand it to a two-same-word *GP*, and the path length is set to 1.

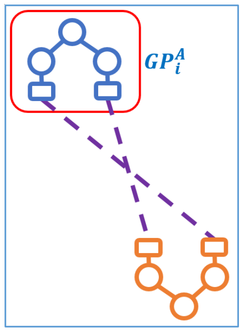


Figure 3.2: An example of a *minimum basic cycle*. The red rectangle shows the *GP* of A to be extracted. The orange *GP* belongs to the other document in the document comparison.

The running time of **Algorithm 3.1** depends on the size of the set of background documents (i.e. $|\mathbb{B}|$), the lengths of the background documents and the length of the input document A. Since *GPs* are extracted by applying *DSCTP*, then for each document comparison between A and a background document, the running time of this comparison is determined by the running time of *DSCTP* which has been discussed in Section 2.5. Thus, the running time of **Algorithm 3.1** is $O(T_{\text{DSCTP}}(A, b_i)|\mathbb{B}|)$, where T_{DSCTP} denotes the running time of *DSCTP*.

3.4 Abstract Phrase Vector Construction

In this section we provide the details of our *APV-GPC* method. This method takes as input the collection of *GPs* extracted from an input document, and has three stages: first, construct *APs* (i.e. dimensions) for the *APV*; second, classify each extracted input *GP* into the *APs*; and third, compute a significance value for each *AP*. In the first stage, naturally utilize a background document set, and apply *GP-DSCTP* to extract *GPs* from the background documents followed by a clustering of these *GPs*. This *GP* clustering will be a hard clustering[4], which means that the *GPs* in each cluster will be semantically similar, hence each *GP* cluster forms an *AP*. In the second stage, each *GP* extracted from the input document is compared (semantically) to each *AP*. If there is at least one *GP* in an *AP* which is semantically similar to the input *GP*, then the input *GP* is *classified* to this cluster. Note that an input *GP* can be classified to multiple clusters. Finally, in the third stage, for each classification of an input *GP*, based on the semantic similarity between this *GP* and each similar *GP* in the corresponding cluster, and on their path lengths, we compute a real value as a vector value for the *APV* reflecting how much of the semantics of the input document are captured by this cluster. A procedure for *APV-GPC* is illustrated in Figure 3.3. The three stages, ***AP*-construction**, ***GP*-classification** and ***AP*-significance-assignment**, are detailed in Sections 3.4.1, 3.4.2, and 3.4.3.

3.4.1 Abstract Phrase Construction

The first major stage of our *APV* construction method is *AP*-construction. The core task of the *AP*-construction is performing a hard clustering over a set of *GPs* extracted from the background documents. To address this task, a recursive clustering method based on *spectral clustering*[5] [105] is utilized. The general idea of this method is first, extracting *GPs* from the background documents and constructing a graph over those *GPs*; and second, recursively partitioning this graph into a set of subgraphs

[4] Each *GP* can belong to only one cluster.

[5] Intuitively, spectral clustering can be understood as solving a minimum graph cut problem with respect to a given number of cuts. Spectral clustering requires a Laplacian matrix of the input graph, where the weights of the graph are similarities between vertices. Typically, a normalized Laplacian matrix is used to obtain balanced-sized clusters. The formalism and a detailed explanation of spectral clustering can be found in [157].

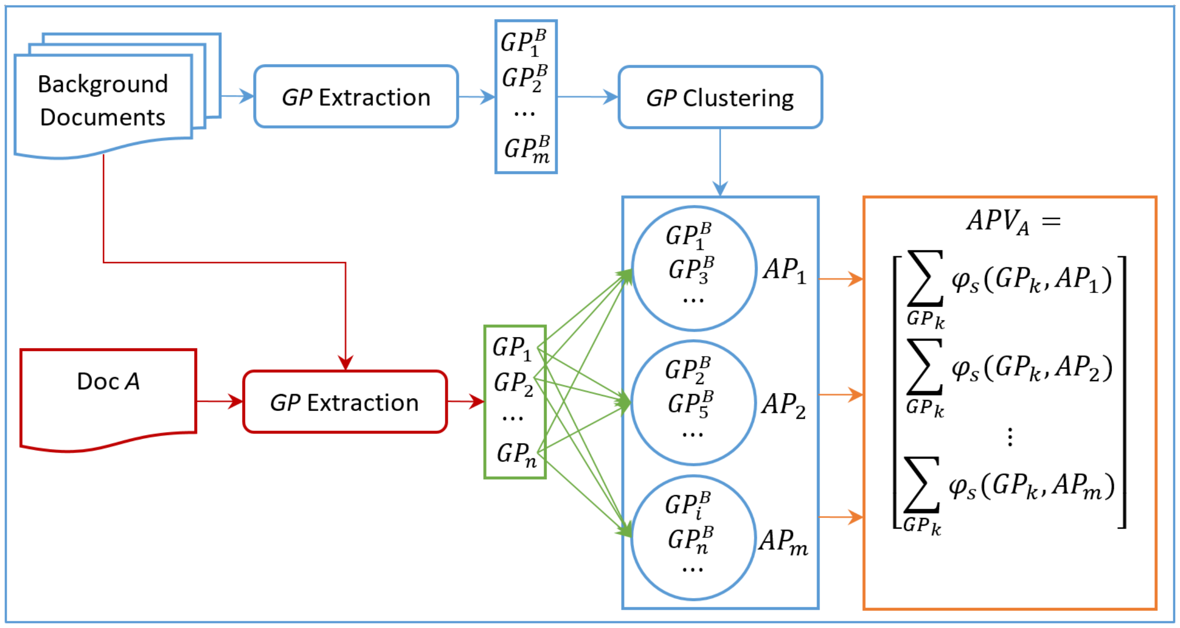

Background Documents
GP Extraction
GP_1^B GP_2^B … GP_m^B
GP Clustering
Doc A
GP Extraction
GP_1 GP_2 … GP_n
GP_1^B GP_3^B …
AP_1
GP_2^B GP_5^B …
AP_2
GP_i^B GP_n^B …
AP_m
$APV_A = \begin{bmatrix} \sum_{GP_k} \varphi_s(GP_k, AP_1) \\ \sum_{GP_k} \varphi_s(GP_k, AP_2) \\ \vdots \\ \sum_{GP_k} \varphi_s(GP_k, AP_m) \end{bmatrix}$

Figure 3.3: *APV-GPC* is diagrammed here. The blue pipeline shows the *GP*-clustering-based *AP*-construction. GP_i^B's are extracted *GPs* from the background documents, and AP_l's are *APs*. The red pipeline is the *GP*-extraction for the input document *A*. GP_j's are extracted *GPs* from *A*. The green pipeline shows the *GP*-classification stage. Each GP_j is classified to one or multiple AP_l's. The orange pipeline shows the *AP*-significance-assignment stage. APV_A denotes the *APV* for the document *A*, and φ_s denotes the *AP*-significance-assignment for each extracted *GP* and an *AP*.

until each subgraph's *diameter*[6] is no more than a specific threshold. This threshold indicates how dissimilar the *GPs* in a cluster may be. Then, each resulting subgraph is a cluster. To construct the graph over the *GPs* in the first step, the similarity value between any two *GPs* is required. We compute this value based on the similarities between words in the two *GPs*. Finally, each resulting cluster of *GPs* is taken as an *AP*. The *AP*-construction method is detailed in **Algorithm 3.2**.

Algorithm 3.2:

Given:

[6] A *diameter* is the length of the longest shortest path in a graph

- A set $\mathbb{B}$ of *background documents.*
- A threshold θ_{dia} for graph diameters.
- A threshold θ_{GP} for *GP* similarities.

Seek:

A set of *APs*, each of which is a cluster of *GPs* extracted from the background documents in $\mathbb{B}$.

***GP* *Graph Construction*:**

```
LET $\mathbb{GP}_{\mathbb{B}}$ denote the set of GPs extracted from the background documents. Initially, $\mathbb{GP}_{\mathbb{B}}$ is empty.
LET $\mathcal{G}_{\mathbb{B}} = (\mathcal{V}_{\mathbb{B}}, \mathcal{E}_{\mathbb{B}})$ denote the graph constructed over $\mathbb{GP}_{\mathbb{B}}$. Initially, $\mathcal{G}_{\mathbb{B}}$ is empty.
FOR $b_i \in \mathbb{B}$:
    Apply Algorithm 3.1 taking $b_i$ and $\mathbb{B} \setminus \{b_i\}$ as inputs;
    LET $\mathbb{GP}_{b_i}$ denote the set of extracted GPs returned from Algorithm 3.1;
    LET $\mathbb{GP}_{\mathbb{B}} = \mathbb{GP}_{\mathbb{B}} \cup \mathbb{GP}_{b_i}$;
FOR $GP_j^{\mathbb{B}} = \{w_{j1}^{\mathbb{B}}, w_{j2}^{\mathbb{B}}\} \in \mathbb{GP}_{\mathbb{B}}$:
    IF $GP_j^{\mathbb{B}} \notin \mathcal{V}_{\mathbb{B}}$:
        Add $GP_j^{\mathbb{B}}$ as a vertex to $\mathcal{G}_{\mathbb{B}}$;
    FOR $GP_k^{\mathbb{B}} = \{w_{k1}^{\mathbb{B}}, w_{k2}^{\mathbb{B}}\} \neq GP_j^{\mathbb{B}} \in \mathbb{GP}_{\mathbb{B}}$:
        IF $GP_k^{\mathbb{B}} \notin \mathcal{V}_{\mathbb{B}}$:
            Add $GP_k^{\mathbb{B}}$ as a vertex to $\mathcal{G}_{\mathbb{B}}$;
        IF $(GP_j^{\mathbb{B}}, GP_k^{\mathbb{B}}) \notin \mathcal{E}_{\mathbb{B}}$:
            Add $(GP_j^{\mathbb{B}}, GP_k^{\mathbb{B}})$ as an edge to $\mathcal{G}_{\mathbb{B}}$;
        LET $Sim_w$ denote the word similarity function used in DSCTP;
        LET $Sim_{GP}$ denote a $GP$ similarity function defined as:

        $Sim_{GP}(GP_j^{\mathbb{B}}, GP_k^{\mathbb{B}}) = \max\{\min\{Sim_w(w_{j1}^{\mathbb{B}}, w_{k1}^{\mathbb{B}}), Sim_w(w_{j2}^{\mathbb{B}}, w_{k2}^{\mathbb{B}})\}, \min\{Sim_w(w_{j1}^{\mathbb{B}}, w_{k2}^{\mathbb{B}}), Sim_w(w_{j2}^{\mathbb{B}}, w_{k1}^{\mathbb{B}})\}\}$

        $w_{GP}(GP_j^{\mathbb{B}}, GP_k^{\mathbb{B}}) = \begin{cases} Sim_{GP}(GP_j^{\mathbb{B}}, GP_k^{\mathbb{B}}) & \text{if } Sim_{GP}(GP_j^{\mathbb{B}}, GP_k^{\mathbb{B}}) \geq \theta_{GP} \\ 0 & \text{otherwise} \end{cases}$

        Set the weight on $(GP_j^{\mathbb{B}}, GP_k^{\mathbb{B}})$ to $w_{GP}(GP_j^{\mathbb{B}}, GP_k^{\mathbb{B}})$;
Recursive Clustering:
LET $\mathbb{C}_{GP}$ denote the set of clusters of GPs extracted from $\mathbb{GP}_{\mathbb{B}}$. Initially, $\mathbb{C}_{GP} = \emptyset$.
LET $\mathbb{G}_{\mathbb{B}}^{C}$ denote the set of connected subgraphs in $\mathcal{G}_{\mathbb{B}}$.
LET Diameter : $\mathbb{G}_{\mathbb{B}}^{C} \to \mathbb{R}_{\geq 0}$ denote the diameter of a subgraph. Diameter is a function.
LABEL_RECURSION:
FOR $\mathcal{G}_{\mathbb{B}}^{i} = (\mathcal{V}_{\mathbb{B}}^{i}, \mathcal{E}_{\mathbb{B}}^{i}) \in \mathbb{G}_{\mathbb{B}}^{C}$:
    LET $\mathbb{G}_{\mathbb{B}}^{C} = \mathbb{G}_{\mathbb{B}}^{C} \setminus \mathcal{G}_{\mathbb{B}}^{i}$;
    IF Diameter($\mathcal{G}_{\mathbb{B}}^{i}$) $\leq \theta_{dia}$:
        LET $\mathbb{C}_{GP} = \mathbb{C}_{GP} \cup \{\mathcal{V}_{\mathbb{B}}^{i}\}$;
    ELSE:
        LET $d_{avg}^{\mathcal{G}_{\mathbb{B}}^{i}}$ denote the average degree of $\mathcal{G}_{\mathbb{B}}^{i}$;
        LET $N_{pred}^{\mathcal{G}_{\mathbb{B}}^{i}}$ denote the target number of clusters for $\mathcal{G}_{\mathbb{B}}^{i}$, computed as:

            $N_{pred}^{\mathcal{G}_{\mathbb{B}}^{i}} = \frac{|\mathcal{V}_{\mathbb{B}}^{i}|}{\sum_{j=0}^{\theta_{dia}} (d_{avg}^{\mathcal{G}_{\mathbb{B}}^{i}})^{j}}$

        IF $N_{pred}^{\mathcal{G}_{\mathbb{B}}^{i}} < 2$:
            **LET** $N_{pred}^{\mathcal{G}_{\mathbb{B}}^i} = 2$;
        Apply *spectral clustering* over $\mathcal{G}_{\mathbb{B}}^i$ with normalized *Laplacian*, and the number of clusters set to $N_{pred}^{\mathcal{G}_{\mathbb{B}}^i}$;
        **LET** $\mathbb{C}_{\mathcal{G}_{\mathbb{B}}^i}$ denote the set of resulting clusters (i.e. subgraphs) for $\mathcal{G}_{\mathbb{B}}^i$;
        **LET** $\mathbb{G}_{\mathbb{B}}^C = \mathbb{G}_{\mathbb{B}}^C \cup \mathbb{C}_{\mathcal{G}_{\mathbb{B}}^i}$;
**IF** $\mathbb{G}_{\mathbb{B}}^C \neq \emptyset$:
    **GOTO LABEL_RECURSION**
***Output APs*:**
**FOR** $\mathcal{C}_{GP}^i \in \mathbb{C}_{GP}$:
    **LET** $\mathbb{GP}_k^{\mathbb{B}} \subseteq \mathcal{C}_{GP}^i$ denote a sub-collection of *GPs* in $\mathcal{C}_{GP}^i$ that share the same two words[7] $\{w_{k1}^{\mathbb{B}}, w_{k2}^{\mathbb{B}}\}$.
    **LET** $\mathcal{C}_{GP}^i = \mathcal{C}_{GP}^i \setminus \mathbb{GP}_k^{\mathbb{B}}$;
    Define a new *GP*: $\hat{GP}_k^{\mathbb{B}} = (\{w_{k1}^{\mathbb{B}}, w_{k2}^{\mathbb{B}}\}, PathLen_{\hat{GP}_k^{\mathbb{B}}})$, where

        $$PathLen_{\hat{GP}_k^{\mathbb{B}}} = \frac{\sum_{GP_{kj}^{\mathbb{B}} \in \mathbb{GP}_k^{\mathbb{B}}} PathLen_{GP_{kj}^{\mathbb{B}}}}{|\mathbb{GP}_k^{\mathbb{B}}|}$$

    **LET** $\mathcal{C}_{GP}^i = \mathcal{C}_{GP}^i \cup \{\hat{GP}_k^{\mathbb{B}}\}$;
**RETURN** $\mathbb{C}_{GP}$, where each $\mathcal{C}_{GP}^i \in \mathbb{C}_{GP}$ is an *AP*.
```

Note that in **Algorithm 3.2** the threshold for *GP* similarities is set to the same value as the word similarity threshold used in *DSCTP*. The reason is as follows: In *DSCTP*, given two *GPs* in two documents, $G_1 = \{w_1^a, w_1^b\}$ and $G_2 = \{w_2^a, w_2^b\}$, there are only two ways to pair the four words (i.e. $\{(w_1^a, w_2^a), (w_1^b, w_2^b)\}$ and $\{(w_1^a, w_2^b), (w_1^b, w_2^a)\}$). For either of these two ways, only when both pairs of words (e.g. (w_1^a, w_2^a) and (w_1^b, w_2^b)) have similarities greater than the word similarity threshold, will these two pairs form

[7] One-word *GPs* also exist, but without losing of generality, for convenience, we consider two-word *GPs*.

a *minimum basic cycle*. We say in this case that G_1 and G_2 are *semantically similar*. Hence, Sim_{GP} is bounded by Sim_w, and the threshold for Sim_{GP} is set to the threshold for Sim_w.

In the recursive clustering, for each clustering iteration, we predict the number of clusters $N_{pred}^{\mathcal{G}_{\mathbb{B}}^i}$ required by *spectral clustering*. This is an empirical method specific to our *GP* clustering. In our case, a typical *GP* graph consists of an overwhelmingly large component and a set of small components. As an example, we sampled 500 documents from the *20Newsgroup* [78] dataset, and applied the *DSCTP*-based *GP*-extraction method to extract 71640 *GPs*. Then we constructed a *GP* graph over these *GPs*. In this graph, there are 623 components. The largest component has 70285 *GPs*, and the sizes of the remaining components range from 1 to 8. In the largest component, the average degree is 13 (with standard deviation 18), and the diameter of this component is 19. For such a graph, classic and state-of-the-art methods for determining the number of clusters would either be ineffective (e.g. *elbow* [73] and *eigengap* [37]) or inefficient (e.g. *self-tuning spectral clustering* [169] and *"jump" inspired by rate distortion theory* [146])[8]. Thus, as seen in the algorithm, we provide a method to determine the number of clusters to be input to the spectral clustering. Specifically, at each recursion, we obtain the average degree for the input graph, and compute the size of an average subgraph where each of its vertices is of the average degree. Then the maximal number of such subgraphs disjointly contained in the input graph is our target number of clusters.

Another alternative recursive clustering is to use bipartite clustering (i.e. always set $N_{pred}^{\mathcal{G}_{\mathbb{B}}^i}$ to 2). However, bipartite recursive clustering is considerably slower than using a target number of clusters, and it may not produce a better performance. In Chapter 5, this will be seen in the experimental results. To distinguish these two options, we call the recursive spectral clustering with target number of clusters **TRSC**, and call

[8] Methods that do not require multiple clustering trials such as *eigengap* could hardly be effective because it is difficult to infer appropriate "cut points" from a graph that does not have desired characteristics implying explicit clusters. Methods that do require multiple trials such as *self-tuning spectral clustering* and *"jump"* are far from efficient.

the bipartite recursive spectral clustering **BRSC**.

Next, we give an analysis of the running time of **Algorithm 3.2**.

The running time of the ***GP Graph Construction*** stage is quadratic to the total number $|\mathbb{GP}_{\mathbb{B}}|$ of *GPs*.

The running time of the ***Recursive Clustering*** stage is $O(\sum_i T_{SC}(N_i))$, where N_i denotes the size of the subgraph as the input to spectral clustering at the *ith* iteration ($i \leq |\mathbb{GP}_{\mathbb{B}}|$), T_{SC} denotes the running time of spectral clustering. In the worst case, the total number of iterations of the *Recursive Clustering* is $O(|\mathbb{GP}_{\mathbb{B}}|)$ which makes it behave in a *decrease-and-conquer* manner (i.e. at each iteration, only a constant number of *GPs* are separated from the input subgraph and are put into some clusters). Then, in this worse case, spectral clustering will be run for $O(|\mathbb{GP}_{\mathbb{B}}|^2)$ times. On the other hand, generally, spectral clustering consists of two steps [157]: first, space transform by solving an eigenvector problem; second, cluster data points in the transformed space (e.g. using k-means). The first step typically[9] costs $O(N_i^3)$, and the second step, by using k-means with *Lloyd's* algorithm [58], typically costs $O(N_i d K_i j)$, where d is the dimension of eigenvectors obtained in the first step ($d \leq N_i$), K_i is the desired number of clusters at the *ith* iteration, and j is the number of iterations specified for convergence (or termination) of k-means. Thus, for each iteration, spectral clustering costs $O(N_i^3 + N_i d K_i j)$, and the total running time of this stage is $O(\sum_i (N_i^3 + N_i d K_i j))$.

The running time of the ***Output APs*** stage costs $O(|\mathbb{GP}_{\mathbb{B}}|)$.

Hence, the running time of *Algorithm 3.2* is dominated by the ***Recursive Clustering*** stage, and is $O(\sum_i (N_i^3 + N_i d K_i j))$.

3.4.2 Generalized Phrase Classification

The second major stage in *APV-GPC* is the *GP*-classification. Specifically, given a set of *GPs* extracted from the input document, and a set of *APs* obtained from the background document set, each extracted *GP* is classified to some number

[9] Faster solutions for the eigenvector problem exist [39].

of *APs*. As long as there is at least one *GP* in an *AP* that is semantically similar to an extracted *GP*, then the extracted *GP* is classified to that *AP*. The procedure for *GP*-classification is stated in **Algorithm 3.3**.

Algorithm 3.3:

Given:

- A set $\mathbb{GP}_A$ of *GPs* extracted from a given document *A*.
- A set $\mathbb{AP}$ of *APs* extracted from the background document set.
- A threshold θ_{GP} for *GP* similarities.

Seek:

A classification of each GP_i^A in $\mathbb{GP}_A$ to the AP_k's in $\mathbb{AP}$ if any.

```
LET $\mathbb{S}_{\mathbb{AP}}$ denote an array recording the resulting classifications of GPs in $\mathbb{GP}_A$,
and $\mathbb{S}_{\mathbb{AP}}$ is indexed by $\mathbb{AP}$. Initially, $\mathbb{S}_{\mathbb{AP}}$ is empty.
FOR $GP_i^A \in \mathbb{GP}_A$:
    FOR $AP_k \in \mathbb{AP}$:
        FOR $\hat{GP}_j^{AP_k} \in AP_k$:
            IF $Sim_{GP}(GP_i^A, \hat{GP}_j^{AP_k}) \geq \theta_{GP}$:
                $\mathbb{S}_{\mathbb{AP}}[AP_k] =$
                $\mathbb{S}_{\mathbb{AP}}[AP_k] \cup \{(GP_i^A, \hat{GP}_j^{AP_k}, Sim_{GP}(GP_i^A, \hat{GP}_j^{AP_k}))\}$
RETURN $\mathbb{S}_{\mathbb{AP}}$
```

Note that the running time of **Algorithm 3.3** is $O(|\mathbb{GP}_A|N_{\mathbb{AP}})$, where $N_{\mathbb{AP}}$ denotes the total number of *GPs* contained in $\mathbb{AP}$.

3.4.3 Abstract Phrase Significance Assignment

The final stage of *APV-GPC* is *AP*-significance-assignment. The significance value for each *AP* is a vector value for the *APV*. This significance value is computed based on the *GP* similarities for an *AP* as obtained in the previous stage and the corresponding *GPs*' path lengths. Recall that the path lengths of *GPs* are used by *DSCTP* in computing the significance of *minimum basic cycles.* Intuitively, a path length reflects how concretely meaningful a *GP* is. The shorter the path length, the more likely that the *GP* has a common-sense meaning. Our *AP*-significance-assignment method is detailed in **Algorithm 3.4**.

Algorithm 3.4:

Given:

- A set $\mathbb{GP}_A$ of *GPs* extracted from a given document *A*.
- A set $\mathbb{AP}$ of *APs* extracted from the background document set.
- The *GP* similarities $\mathbb{S}_{\mathbb{AP}}$ obtained in **Algorithm 3.3**.
- A function $w_s : (PathLen_x, PathLen_y) \mapsto s \in \mathbb{R}_{\geq 0}$ computes a significance weight between two *GPs* (e.g. x and y) based on their path lengths $PathLen_x$ and $PathLen_y$. We define w_s to be the same as the significance weight function used in *DSCTP* **Stage 6** (i.e. $w_s(PathLen_x, PathLen_y) = \frac{3}{e^{PathLen_x{}^3} + e^{PathLen_y{}^3}}$).

Seek:

An APV for the document A.

LET APV_A denote the APV for A. Each dimension of APV_A corresponds to an AP in $\mathbb{AP}$. Initially, $APV_A = [0]$.

FOR $AP_k \in \mathbb{AP}$:

```
    FOR $(GP_i^A, \hat{GP}_j^{AP_k}, Sim_{GP}(GP_i^A, \hat{GP}_j^{AP_k})) \in \mathbb{S}_{\mathbb{AP}}[AP_k]$:

        $APV_A[AP_k] =$
        $APV_A[AP_k] + w_s(PathLen_{GP_i^A}, PathLen_{\hat{GP}_j^{AP_k}}) \cdot Sim_{GP}(GP_i^A, \hat{GP}_j^{AP_k})$

RETURN $APV_A$.
```

3.5 Discussion

APVs have been tested in a set of experiments, and the experimentation is explained in Chapter 5.Therein, the empirical running time of *APV-GPC* will also be discussed. We note here that *APV-GPC* does have a couple of intrinsic issues. These issues motivate our work in Chapter 4. Therein, we will discuss the issues and some ideas for improvement.

Chapter 4
GRAPH SIGNAL PROCESSING ON GENERALIZED PHRASE GRAPHS

4.1 Introduction

As mentioned by the end of Chapter 3, *APV-GPC* has two intrinsic issues, and these two issues motivate a new *DSR*. In this section, we first outline these two issues, and then our ideas to address the issues.

The first issue is that the *GP* clustering may force semantically similar *GPs* to be in different clusters, which leads to overlapping of clusters and accordingly impacts the quality of *GP* classification. We call this issue **confused clustering**. We show an example in Figure 4.1. Therein, we suppose that A, B, C and D are four *GPs*. These *GPs* form a *GP* graph, and the edges are created by their semantic similarity relationships. Also, we suppose that, for each cluster, the maximum allowed diameter is 2. Then there are three possible clusterings, each of which performs a cut on the graph. However, either of the three different cuts will separate a pair of semantically similar *GPs*. Moreover, this issue is particularly critical in *APV-GPC*. In the example of *20Newsgroup* as discussed in Section 3.4.1. The average number of *GPs* in an *AP* is 4. It means that every separation of semantically similar *GPs* can lead to a serious degradation of the significance value of an *AP*. This issue is not specific to the *GP* clustering methods (i.e. *BRSC* and *TRSC*) used in *APV-GPC*. Utilizing other hard clustering methods will not solve this issue, because any such method has to confront the situation exemplified in Section 3.4.1 (i.e. a large connected subgraph of *GPs*) and forces some semantically similar *GPs* to be in different clusters.

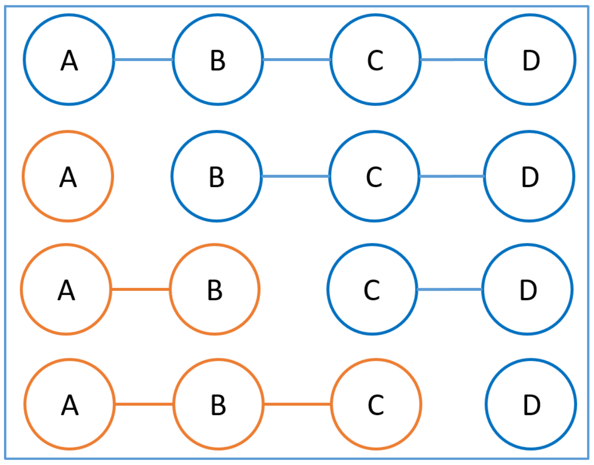


Figure 4.1: An example of *confused clustering*. The first row is a GP graph. The remaining three rows are three different clusterings. However, it is difficult to determine which clustering is the best.

The second issue is that a small change in the GP similarity threshold may lead to significant changes in GP classification. That is, when computing significance values of APs, varying the GP similarity threshold can result in a dramatic change in the significance value contributed by a member GP of an AP with respect to a GP classified to that AP, because the GP similarity will be changed from slightly above the threshold to slightly below the threshold. Neither removing or adjusting the word similarity threshold will solve this issue. Consequently, the significance of an AP may be underestimated. We name this issue **slashed classification**. An example of this issue is shown in Figure 4.2.

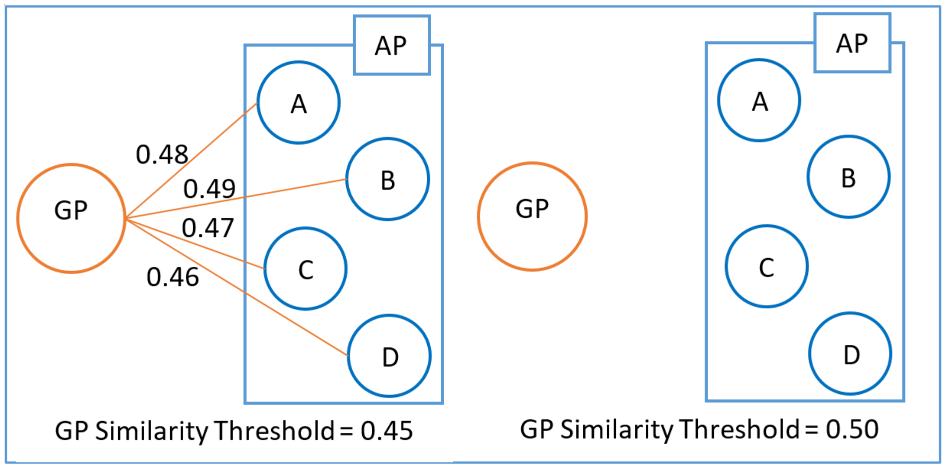

AP
A
B
C
D
GP
0.48
0.49
0.47
0.46
GP Similarity Threshold = 0.45
AP
A
B
C
D
GP
GP Similarity Threshold = 0.50

Figure 4.2: The left and the right figures show two cases of classifying a *GP*. In the left figure, the word similarity threshold is set to 0.45. In this case, the *GP* is classified to the *AP*, and each member of the *AP* (i.e. *A*, *B*, *C* and *D*) has a non-zero semantic similarity value with the *GP*. Those semantic similarity values will then contribute to the significance value of this *AP*. In the right figure, raising the word similarity threshold by merely 0.05 leads to the *GP* not being classified to the *AP*. In this case, the significance value of the *AP* will be zero.

In this chapter, we concentrate on finding an improved document semantic representation that overcomes these two drawbacks.

A straightforward approach to address the first issue is to simply not do the *GP* clustering, which means that *APs* are not constructed. Instead, we directly use *GPs* extracted from the background documents to represent the dimensions of *APVs*. The rest of *APV-GPC* remains the same. Thus the output of this modified *APV-GPC* is still a vector, and also a semantic representation for the input document.

To address the second issue, it needs to avoid dramatic changes in *GP* classification caused by small changes of the *GP* similarity threshold. As discussed above, removing or adjusting the *GP* similarity threshold will not solve this issue. An alternative thought is to design a more sophisticated significance assignment method to compute vector values.

4.2 Toward An Improved Significance Assignment Idea

To do that, we take advantage of *GP* graphs. Recall that a *GP graph* is built upon the collection of *GPs* extracted from a set of background documents (Chapter 3.4.1). Each vertex is a *GP*, and the weight on each edge is the semantic similarity between the two endpoint *GPs*. This weight is zero if the similarity is less than the threshold. Such a graph reflects the "strong" similarity relationships between *GPs*. To explain an important observation on *GP* graphs, we define the ***h* nearest neighbor graph (*h*-NNG)**:

Def: *h* nearest neighbor graph (*h*-NNG):

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph. For a vertex $v_i \in \mathcal{V}$, a vertex $v_j \neq v_i \in \mathcal{V}$ is a ***k*-hop neighbor** of v_i if there is a shortest path between v_i and v_j and the length of the shortest path is an integer $k > 0$. The ***h* nearest neighbor graph** of v_i is the subgraph of $\mathcal{G}$ induced by v_i and all of its *k-hop neighbors* for $k \in [1, h]$.

We observe that in the *h-NNG* of a vertex in a *GP* graph if h is low[1] (e.g. 2), then all *k-hop neighbors* ($k \leq h$) of the vertex have relatively high similarities to the vertex. Also, if a *GP* from the input document is semantically similar to a vertex in a *GP* graph, then it is likely that the input *GP* has relatively high similarities to the *k-hop neighbors* of the vertex when k is bounded by a low number h. The lower the bound h and the higher the weights in the *h-NNG* of the vertex, the more likely that the input *GP* has relatively high similarities to the *k-hop neighbors* of the vertex.

Based on this observation, we propose the following idea for significance assignment. Let GP_i be a *GP* extracted from an input document, let $GP_j^{\mathbb{B}}$ be a vertex in the *GP* graph constructed on a set of background documents, and let $\mathcal{G}^h_{GP_j^{\mathbb{B}}}$ be the *h-NNG* of $GP_j^{\mathbb{B}}$. Suppose that GP_i is semantically similar to $GP_j^{\mathbb{B}}$. Then the significance value

[1] In the example discussed in Section 3.4.1, the average diameter of the largest component in the *GP* graph constructed over the *GPs* extracted from the documents is 19.

for $GP_j^{\mathbb{B}}$ with respect to GP_i is computed. In addition, we compute a significance value for each k-hop neighbor of $GP_j^{\mathbb{B}}$ (for $j \leq h$). This computed value is determined by the significance value of $GP_j^{\mathbb{B}}$ and the similarity between the k-hop neighbor and $GP_j^{\mathbb{B}}$. We call this value the **induced significance value**[2] for the k-hop neighbor of $GP_j^{\mathbb{B}}$. We impose two design guidelines for the induced significance values.

C1. The induced significance values for the k-hop neighbors of $GP_k^{\mathbb{B}}$ should be significantly lower than the significance value of $GP_k^{\mathbb{B}}$.

C2. k-hop neighbors of higher k should have lower induced values than those of lower k.

Next, we consider the effectiveness of such a significance assignment meeting **C1** and **C2**. The induced significance values for the k-hop neighbors of $GP_k^{\mathbb{B}}$ are key. Because of **C1**, the induced values will not overwhelm the significance of $GP_k^{\mathbb{B}}$ if that value is high. More importantly, the induced significance values for the k-hop neighbors can compensate for the dramatic changes in significance values caused by the GP similarity threshold. To see this, suppose that given another input GP GP_j, $0 < \theta_{GP} - Sim_{GP}(GP_j, GP_k^{\mathbb{B}}) \leq \epsilon_{GP}$, where θ_{GP} is the GP similarity threshold, and $0 < \epsilon_{GP} \ll \theta_{GP}$. In this case, the significance assignment method proposed in Section 3.4.3 will produce zero as the significance value for $GP_k^{\mathbb{B}}$ with respect to GP_j, which is a dramatic change in the significance even though GP_j is not indisputably dissimilar to $GP_k^{\mathbb{B}}$. As long as GP_j is semantically similar (by the threshold) to at least one k-hop neighbor of $GP_k^{\mathbb{B}}$, then the induced significance values for $GP_k^{\mathbb{B}}$ will at least partially compensate for the dramatic change at $GP_k^{\mathbb{B}}$.

This idea is demonstrated by an example in Figure 4.3. This figure consists of four cases depicted in **a**, **b**, **c** and **d**. Specifically, suppose the semantic similarity between an input GP GP_j and another GP $GP_k^{\mathbb{B}}$ is less than, but very close to, the threshold, (i.e. $0 < \theta_{GP} - Sim_{GP}(GP_j, GP_k^{\mathbb{B}}) \leq \epsilon_{GP}$). Figure 4.3 shows the compensation idea above in four cases by considering the 1-*NNG* of $GP_k^{\mathbb{B}}$.

[2] When no confusion is in the context, we use *induced value* for short.

a: GP_j does not have a similarity greater than θ_{GP} with any 1-hop neighbor of $GP_k^{\mathbb{B}}$. Then no induced significance value is assigned to $GP_k^{\mathbb{B}}$, and thus no compensation is made.

b: GP_j is similar to $GP_a^{\mathbb{B}}$. Since $GP_k^{\mathbb{B}}$ and $GP_b^{\mathbb{B}}$ are similar to $GP_a^{\mathbb{B}}$, then in addition to computing a significance value for $GP_a^{\mathbb{B}}$, induced significance values will be computed for $GP_k^{\mathbb{B}}$ and $GP_b^{\mathbb{B}}$. The induced values will be significantly lower than the significance value of $GP_a^{\mathbb{B}}$. In this case, the path "$GP_j - GP_a^{\mathbb{B}} - GP_k^{\mathbb{B}}$" is evidence of possible similarity between GP_j and $GP_k^{\mathbb{B}}$, and thus reasonably $GP_k^{\mathbb{B}}$ gets an induced value that is small, but non-zero.

c: When GP_j is similar to two of $GP_k^{\mathbb{B}}$'s 1-hop neighbors, more induced values are accumulated to $GP_k^{\mathbb{B}}$. Then the significance of $GP_k^{\mathbb{B}}$ is further confirmed.

d: In the case that GP_j is similar to all of the 1-hop neighbors of $GP_k^{\mathbb{B}}$, the significance value of $GP_k^{\mathbb{B}}$, due to the accumulation of three induced values, may be fairly large.

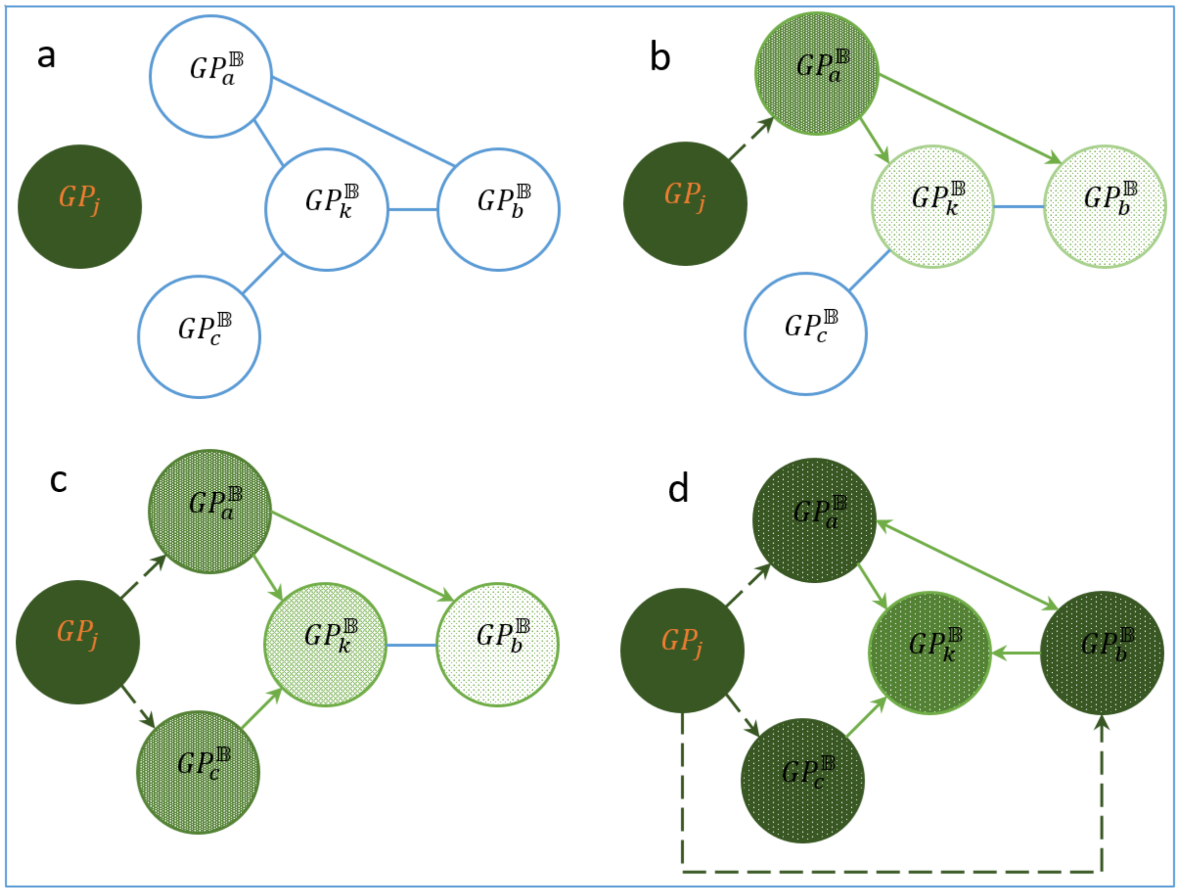


Figure 4.3: GP_j is an input GP. $GP_k^{\mathbb{B}}$ is a vertex in a GP graph. $GP_k^{\mathbb{B}}$, $GP_a^{\mathbb{B}}$, $GP_b^{\mathbb{B}}$ and $GP_c^{\mathbb{B}}$ form the 1-NNG of $GP_k^{\mathbb{B}}$. $GP_a^{\mathbb{B}}$, $GP_b^{\mathbb{B}}$ and $GP_c^{\mathbb{B}}$ are three 1-hop neighbors of $GP_k^{\mathbb{B}}$. Likewise, $GP_a^{\mathbb{B}}$ and $GP_b^{\mathbb{B}}$ are 1-hop neighbors of each other. Also, $GP_a^{\mathbb{B}}$ and $GP_c^{\mathbb{B}}$ are 2-hop neighbors of each other, as are $GP_b^{\mathbb{B}}$ and $GP_c^{\mathbb{B}}$. We suppose that GP_j satisfies $0 < \theta_{GP} - Sim_{GP}(GP_j, GP_k^{\mathbb{B}}) \leq \epsilon_{GP}$.
Legends: A dashed arrow indicates a similarity relationship between GP_j and a vertex. A solid green arrow indicates that a target vertex obtains an induced significance value from the source. Green vertices are the ones that have been assigned significance values while blue ones have not. The darker a green vertex, the higher its significance value. From **a** to **d**, the more 1-hop neighbors of $GP_k^{\mathbb{B}}$ similar to GP_j, the more induced values will be summed to $GP_k^{\mathbb{B}}$.

Assigning significance values to background documents' *GPs* for a document by using induced values has an analogue in the *heat diffusion* [22] process. The *GP* graph built upon the background documents' *GPs* can be imagined as a cold (at zero temperature) and uniform iron sheet. Each *GP* extracted from the input document then will be compared to each vertex of the *GP* graph. If an input *GP* is similar to a *GP* graph vertex, then a significance value is assigned to that vertex. A vertex can be imagined as a spot, and heating up this spot can be imagined as significance value assignment (we call this a *hot spot*). After a certain period of time (we call it a *diffusion period*), the heat at each hot spot will have been diffused in a range centered at this spot. Naturally, the temperatures of the spots within the range then will rise. The nearer a spot to the center, the larger the change of temperature. The k-hop neighbors thus can be imagined as the spots within such a range, and the induced values for the neighbors as the risen temperatures at the spots.

To realize the idea using induced values, we utilize *graph signal processing (GSP)* [128] [135] [129] techniques to compute the induced significance value. These techniques are naturally consonant with the "heat diffusion" intuition. Section 4.3 provides some necessary fundamentals and explanations of *GSP*. The new document semantics construction method based on our improved significance assignment is introduced in Section 4.4.

4.3 Graph Signal Processing in a Nutshell

Graph signal processing has four major fundamental components, *graphs*, *signals*, *Fourier transform* and *filters*. We elaborate below on each of these.

Graphs:

Graphs here are weighted and undirected finite graphs. Self-loops and multi-edges are not allowed. The weight on each edge should reflect the similarity between the two ends, and are naturally non-negative.

Signals:

A **signal** on a graph is a function defined on the set of vertices mapping each vertex to a real value. The empirical meaning of a signal is scenario-dependent. For example, given a *GP* graph $\mathcal{G}$ and an input *GP* extracted from a document, the similarity between the input *GP* and each vertex of $\mathcal{G}$ can be taken as a signal.

Fourier transform:

The **graph Fourier transform** is defined as follows:

Def: Graph Fourier Transform:

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected, weighted and undirected graph with $|\mathcal{V}| = N$. Let L be the Laplacian matrix ($N \times N$) for $\mathcal{G}$. Also, let $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ be the eigenvalues of L, and let $\{g_1, g_2, \ldots, g_N\}$ be the corresponding eigenvectors. Then, for a function $f : \mathcal{V} \to \mathbb{R}$ defined on $\mathcal{G}$, the **graph Fourier transform** of f is a vector, and each vector value is defined by [135]:

$$\hat{f}(\lambda_i) = \langle \boldsymbol{f}, g_i \rangle = \sum_{k=1}^{N} f(k) g_i^*(k) \tag{4.1}$$

where $\boldsymbol{f}$ is the vector form of f and $f(k)$ is the function value for the kth vertex. The λ_i's are called the **frequencies** with respect to the graph Fourier transform.

The **inverse graph Fourier transform** is defined by:

$$f(i) = \sum_{k=1}^{N} \hat{f}(\lambda_k) g_k(i) \tag{4.2}$$

Filters:

The last fundamental component of graph signal processing is *filters*. Applying filters in the graph Fourier transform is similar to the filtering in classic Fourier analysis. The frequency domain filtering for the graph Fourier transform is given by [135]:

$$\hat{f}_{out}(\lambda_k) = \hat{f}_{in}(\lambda_k) \hat{h}(\lambda_k) \tag{4.3}$$

and

$$f_{out}(i) = \sum_{k=1}^{N} \hat{f}_{in}(\lambda_k) \hat{h}(\lambda_k) g_k(i) \tag{4.4}$$

where $\hat{f}_{out}$ is the filtered Fourier transform, and f_{out} is the filtered function.

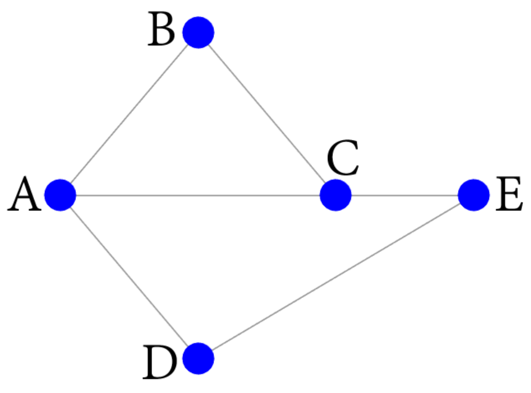


Figure 4.4: A, B, C, D, E are vertices, and all edges are undirected and weighted by 1.

One of the most celebrated filters is the *Heat kernel* low-pass filter, $H(\lambda) = e^{-\tau\lambda}$. $H(\lambda)$ is derived from the fundamental solution to the *Heat equation* in the frequency domain, $\frac{\partial f}{\partial t} = \tau \Delta f$, where τ is the *diffusivity coefficient*[3] and Δ is the Laplacian operator[4]. The curvature of the Heat kernel can be controlled by *tau*. Figures 4.5 and 4.6 show an example of applying the Heat kernel to a graph signal. In Figure 4.5a, an impulse signal[5] on the graph shown in Figure 4.4 is visualized. The signal function takes 1 at the vertex A, and 0 everywhere else. The corresponding graph Fourier transform of this signal is shown in Figure 4.5b. Similar to the classic Fourier transform of an impulse signal, there are many high-frequency components and corresponding non-zero coefficients. We apply the Heat kernel with three different diffusivity coefficients ($tau = 1, 5, 10$) to low-pass filter the signal in the frequency domain. The curves of the Heat kernel with the three *tau*'s are shown in Figure 4.6. The filtered signals and the corresponding graph Fourier transforms are shown in Figure 4.7, in which each

[3] This diffusivity coefficient actually implies both the diffusion period (i.e. time) and the thermal diffusivity.

[4] Readers who are interested in how the Heat kernel is derived and its applications are referred to [29] [170] [11] [55]

[5] An impulse signal on a graph is a function that maps only one vertex to a non-negative real value, and maps the remaining vertices to zero.

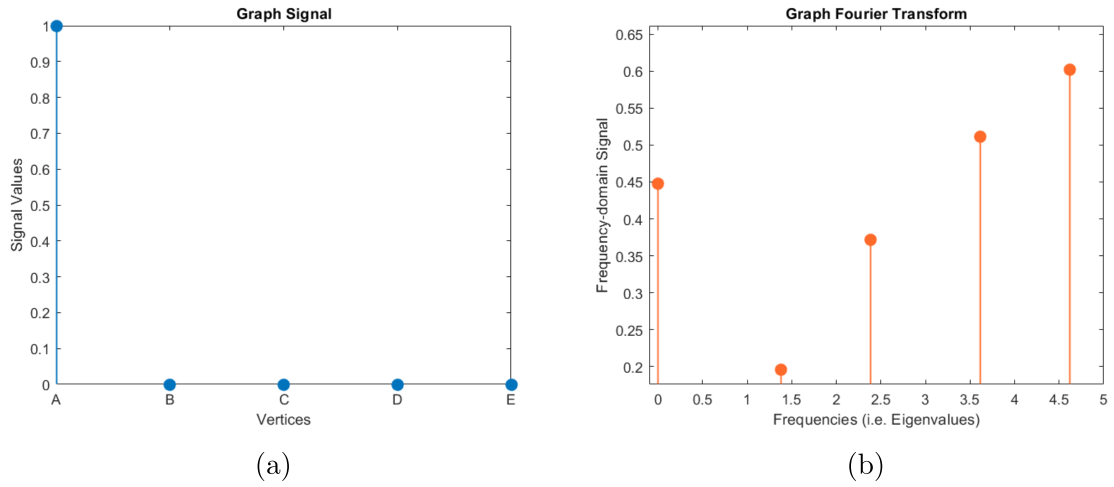


Figure 4.5: Figure **(a)** shows a signal imposed on the graph shown in Figure 4.4. Figure **(b)** shows the graph Fourier transform of the signal.

row visualizes a case with a specific *tau*. These filtering results explicitly demonstrate that the higher the *tau*, the more high-frequency components that are removed (i.e. the smoother the filtered signal). In addition, since the graph Fourier transform is a linear operator, it is feasible to apply a bank of filters to a signal at the same time, and the final filtered signal (called the **synthesized signal**) is the sum of all filtered signals each of which is an output from a distinct filter. Empirically, this is a practical approach to offset the bias in choosing *tau*. In Figure 4.8, we show the synthesized filtered signal and the graph Fourier transform taking advantage of the three filtering results shown in Figure 4.7.

4.4 A GSP-based Document Semantic Representation

In this section, we explain how the improved significance assignment discussed in Section 4.1 is realized. By utilizing *GSP* techniques based on this realization, we introduce a new single document semantic representation, **generalized phrase graph signal (GPGS)**, and the corresponding construction method, **GPGS-GSP**. The *GPGS* is defined as:

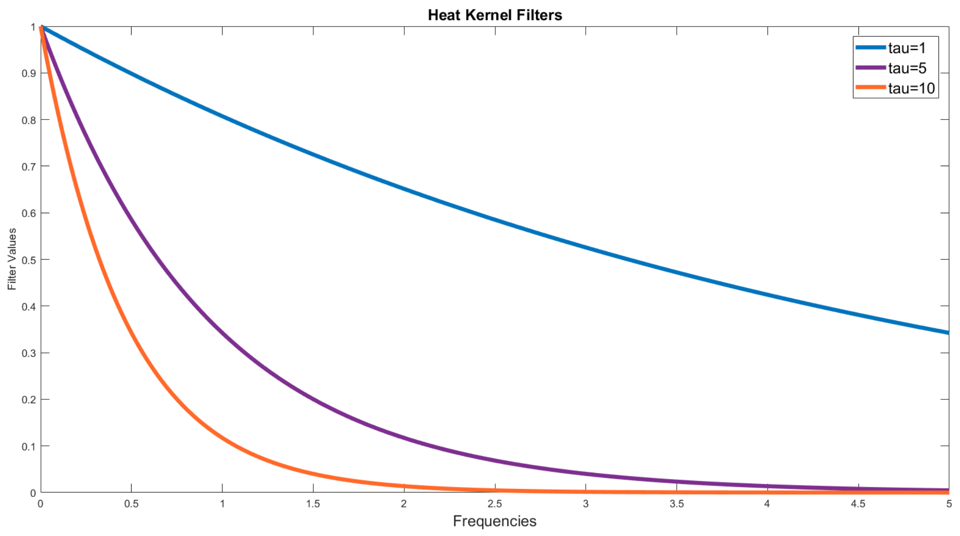
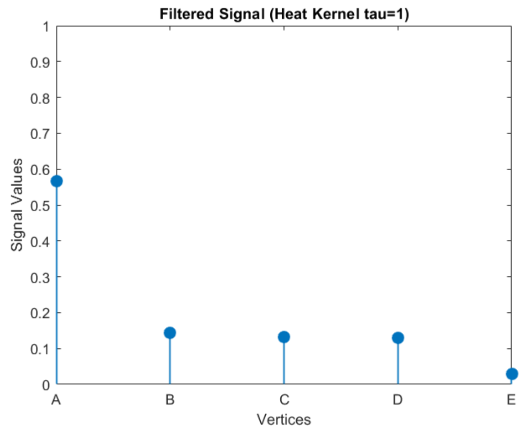


Figure 4.6: The Heat kernels with three different diffusivity coefficients, $tau = 1, 5, 10$.

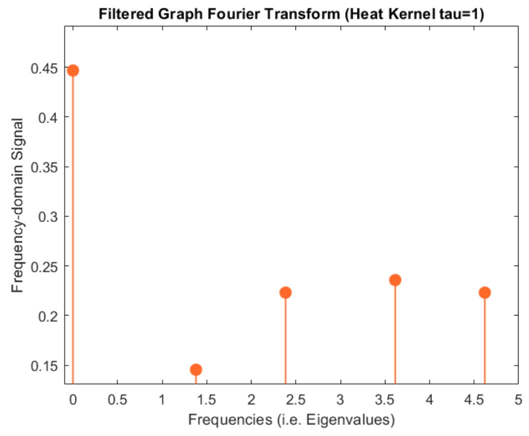
Def: Generalized Phrase Graph Signal (GPGS):

Let $\mathcal{G}_{\mathbb{B}} = (\mathcal{V}_{\mathbb{B}}, \mathcal{E}_{\mathbb{B}})$ be the *GP* graph constructed on a given set $\mathbb{B}$ of background documents. A **GPGS** for a document is a non-negative *signal* on $\mathcal{G}_{\mathbb{B}}$ such that the value at $v_i \in \mathcal{V}_{\mathbb{B}}$ reflects the significance of v_i to the semantics of the document.

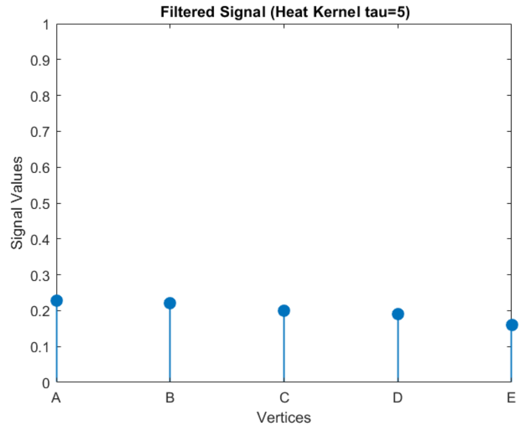
To construct a *GPGS* for a document A, there are four major stages: first, extract *GPs* from the input document; second, construct a *GP* graph for the set of background documents; third, for each vertex v in the *GP* graph and each *GP* in A, assign a significance value to v, and assign an induced significance value to each of v's k-hop neighbors (for k less than or equal to a predetermined value h); and fourth, for each vertex v in the *GP* graph, sum all of its significance values as well as the induced values assigned to v in order to obtain the signal value for the *GPGS* at this vertex. This construction is detailed in **Algorithm 4.1**.



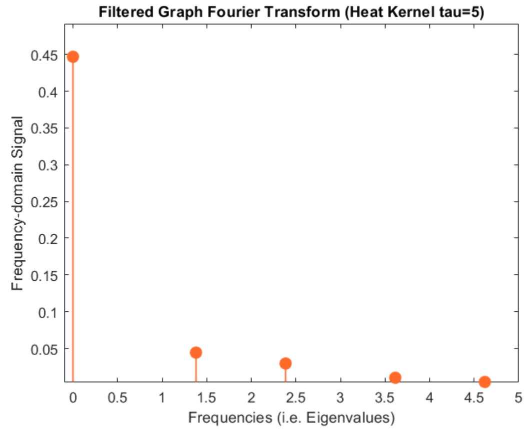
(a)



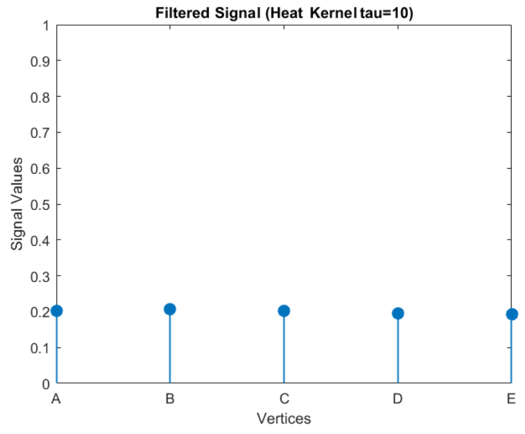
(b)



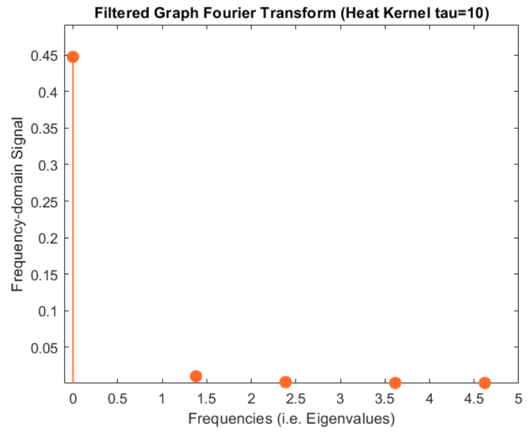
(c)



(d)

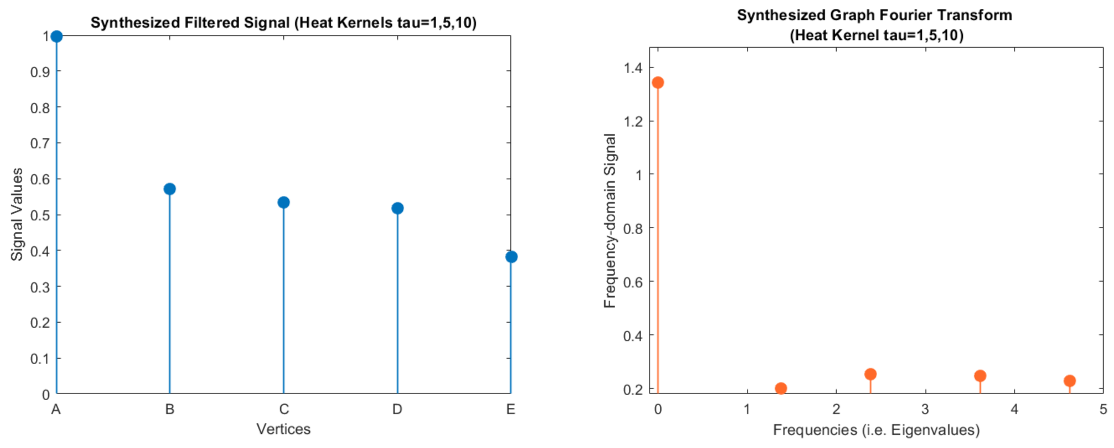


(e)



(f)

Figure 4.7: Filtered signals and their graph Fourier transforms by the Heat kernel with $tau = 1, 5, 10$.



(a) (b)

Figure 4.8: The synthesized filtered signal and the graph Fourier transform based on three filtered signal with $tau = 1, 5, 10$.

Algorithm 4.1:

Given:

- A document A.
- A set $\mathbb{B}$ of background documents.
- A parameter $h > 0$ for h-NNGs.
- A finite set T of tau's ($tau > 0$) for the Heat kernel bank.

Seek:

A *GPGS* for the document A.

GP Extraction:

Apply **Algorithm 3.1** to A to extract a collection of *GPs* from A; **LET** $\mathbb{GP}_A$ denote this *GP* collection.

GP Graph Construction:

Use the *GP* graph construction method in **Algorithm 3.2** to construct a *GP* graph $\mathcal{G}_{\mathbb{B}} = (\mathcal{V}_{\mathbb{B}}, \mathcal{E}_{\mathbb{B}})$.

```
***Significance Value and Induced Value Assignment:***
**FOR** $v_i \in \mathcal{V}_{\mathbb{B}}$:
    Compute $\mathcal{G}^h_{v_i} = (\mathcal{V}^h_{v_i}, \mathcal{E}^h_{v_i})$, the $h$-NNG for $v_i$;
    **FOR** $GP_k \in \mathbb{GP}_A$:
        Construct an impulse signal $\delta_{v_i}$ on $\mathcal{G}^h_{v_i}$ such that
$$\delta_{v_i}(v_l) = \begin{cases} Sim_{GP}(GP_k, v_l) & \text{if } v_l = v_i \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$
        where $v_l \in \mathcal{V}^h_{v_i}$;
        **FOR** $\tau_j \in T$:
            Apply the Heat kernel with $\tau_j$ to $\delta_{v_i}$ by utilizing the graph Fourier transform and the graph signal filtering (as in Section 4.3);
            **LET** $g^{v_i}_{\tau_j}$ denote the filtered signal;
        **LET** $g^{GP_k}_{v_i} = \sum_{\tau_j \in T} g^{v_i}_{\tau_j}$;
            **LET** $s^{v_i}_{GP_k}$ denote a function assigning a significance value to $v_i$ with respect to $GP_k$ and induced significance values to $v_l \in \mathcal{V}^h_{v_i}$ with respect to $GP_k$ such that:
$$s^{v_i}_{GP_k}(v_l) = \begin{cases} Sim_{GP}(GP_k, v_l) \cdot |T| \cdot w_s(PathLen_{GP_k}, PathLen_{v_l}) & \text{if } v_l = v_i \\ g^{GP_k}_{v_i}(v_l) \cdot w_s(PathLen_{GP_k}, PathLen_{v_l}) & \text{otherwise} \end{cases} \tag{4.6}$$
where $w_s$ is the significance weight function used in **Algorithm 3.4**;
    **LET** $s_{v_i}(v_l) = \sum_{GP_k \in \mathbb{GP}_A} s^{v_i}_{GP_k}(v_l)$ for each $v_l \in \mathcal{V}^h_{v_i}$;
***Construct GPGS:***
**FOR** $v_i \in \mathcal{V}_{\mathbb{B}}$:
    Extend $s_{v_i}$ by: $\forall v_l \in \mathcal{V}_{\mathbb{B}}$
```

$$\tilde{s}_{v_i}(v_l) = \begin{cases} s_{v_i}(v_l) & \text{if } s_{v_i}(v_l) \text{ is defined} \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

```
**LET** $GPGS_A(v_l) = \sum_{v_i \in \mathcal{V}_{\mathbb{B}}} \tilde{s}_{v_i}(v_l)$ for each $v_l \in \mathcal{V}_{\mathbb{B}}$
**RETURN** $GPGS_A$.
```

Here, we give some running time analysis of **Algorithm 4.1**.

At the ***GP Extraction*** stage, the running time is determined by the running time of **Algorithm 3.1**, and this running time is proportional to the number of background documents used to extract *GPs* from *A*.

At the ***GP Graph Construction*** stage, the running time is determined by the running time of **Algorithm 3.2**, and this running time is proportional to the number of *GPs* extracted from the background documents.

At the ***Significance Value and Induced Value Assignment*** stage, for each vertex v_i in the *GP* graph, the graph signal filtering will be applied to each *GP* of A. This single step needs to solve an eigenvector problem, which typically costs $O(N_i^3)$, where N_i is the number of vertices in the h-NNG of v_i. This single step will be run for $O(|\mathcal{V}_{\mathbb{B}}||\mathbb{GP}_A|)$ times. Thus, the running time of this stage is $O(\sum_i |\mathcal{V}_{\mathbb{B}}||\mathbb{GP}_A|N_i^3)$.

At the ***Construct GPGS*** stage, the running time is proportional to $O(|\mathcal{V}_{\mathbb{B}}|)$.

Hence, the running time of **Algorithm 4.1** is dominated by the ***Significance Value and Induced Value Assignment*** stage, and is $O(\sum_i |\mathcal{V}_{\mathbb{B}}||\mathbb{GP}_A|N_i^3)$.

4.5 Discussion

When applying **Algorithm 4.1** in our experiments, we use $h = 1$ and $T = \{1, 5, 10\}$. This ultra low-valued h provides tight semantic similarity relationships between a vertex and its k-hop neighbors. As discussed in Section 4.3, using a filter bank instead of only one filter offsets the bias on choosing *tau*.

Another technical detail is that when constructing $s^{v_i}_{GP_k}$, instead of simply using $g^{GP_k}_{v_i}$ to compute the significance values and the induced significance values, we use $Sim_{GP}(GP_k, v_l) \cdot |T|$. The reason is that the signal value at the impulse vertex will decay after the filtering; however, the decayed value will effectively reflect the significance of the impulse vertex. Also, using $Sim_{GP}(GP_k, v_l) \cdot |T|$ keeps the induced significance values significantly lower than the significance value of the impulse vertex, which is consistent with the design guideline **C1** in Section 4.1.

Next, we consider the question: ***when assigning significance values and induced significance values, why do we apply the signal filtering on each h-NNG instead of directly on $\mathcal{G}_{\mathbb{B}}$?*** To answer this question, it is necessary to recall the objective of our improved significance assignment idea. In brief, when assigning the significance value to a *GP*, the semantically similar *GPs* should be assigned induced values to reflect their significance. Thus, by utilizing the graph signal filtering technique, we expected to witness that an impulse signal on a vertex after the filtering would lead to induced values greater than zero at the impulse vertex's k-hop neighbors for $k \leq h$ with a small h (i.e. the filtered signal would rapidly decay along the directions departing away from the impulse vertex). However, it is nearly impossible for this to happen. The underlying reason of the impossibility is the celebrated *uncertainty principle* [60] [72] [160]. Note that in *GSP*, graph harmonic analysis has the same issue [4] [3] [153]. In classic harmonic analysis, the *uncertainty principle* is expressed as $\sigma_t \cdot \sigma_f \geq \frac{1}{4\pi}$, where σ_t and σ_f are the standard deviations of the time and frequency respectively. It implies that a signal cannot be bounded in time and in frequency at the same time. More specifically, if a signal varies rapidly, then it has more high-frequency components; and if we smooth out high-frequency components, then the filtered signal would have to spread widely in time. In [4], the *uncertainty principle* in *GSP* is studied, and it is shown that signals on graphs follow a similar principle. Therein, the **graph spread** about a vertex v_i is defined by $\Delta^2_{g,v_i}(\boldsymbol{f}) = \frac{1}{||\boldsymbol{f}||_2} \sum\limits_{v_k \in \mathcal{V}} d(v_i, v_k)^2 f(v_k)^2$, where $\boldsymbol{f}$ is a graph signal, $d(\cdot, \cdot)$ is a metric on vertices (e.g. the length of shortest path between two vertices), and $\mathcal{V}$ is the set of vertices. The *overall graph spread* is the sum of all

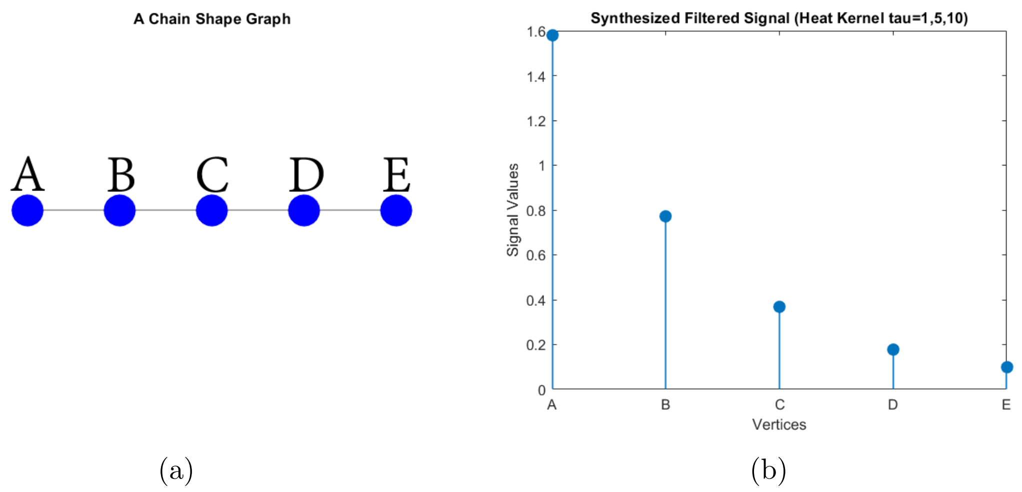


Figure 4.9: An impulse signal centered at A is filtered on a chain shape graph. **(b)** shows the synthesized filtered signal by using the Heat kernel with $tau = 1, 5, 10$. The filtered signal values at D and E are not trivial even though they could hardly be semantically similar to A.

graph spreads about vertices. The **spectral spread** is defined by $\Delta_s^2(\boldsymbol{f}) = \frac{1}{||\boldsymbol{f}||_2}\boldsymbol{f}^T L\boldsymbol{f}$, where L is the Laplacian matrix. The graph spread and the spectral spread are analogues to σ_t and σ_f respectively. Based on the definitions, intuitively, if the spectral spread of an impulse signal is high, then the graph spread is low; on the other hand, if high-components are mostly smoothed, then the spectral spread would decay significantly, but the graph spread has to increase significantly as the filtered signal must not vary rapidly. Thus, in **Algorithm 4.1**, if we apply the signal filtering directly on $\mathcal{G}_{\mathbb{B}}$, then the filtered signal of an impulse must spread in a wide range, which implies that if an input GP is semantically similar to a vertex in the GP graph, then this similarity may suggest that a large number of vertices are also similar to the input GP even though they are distant. Then if we use such filtered signals to assign induced significance values, the design guideline **C2** would be violated. We show an example of this phenomenon in Figure 4.9. To solve this issue, we confine the Laplacian operator instead of the signal values. That is why we have to apply the signal filtering to assign significance and induced values on a h-NNG about a vertex.

We consider the space complexity of **Algorithm 4.1**. Since the GP graph can be quite large, a *GPGS* may require a large amount of space if we store the resulting value for every vertex. One may ask: ***is it possible to make GPGS more space-efficient?*** Empirically, the answer is YES: When implementing **Algorithm 4.1**, only non-zero entries are stored. When two given *GPGS*'s are compared, those *GPGS*'s are extended to the least common vertex set (i.e. the minimum set that contains the vertices in both *GPGS*'s), and the cosine similarity is utilized to compute their similarity value.

To evaluate the performance of *GPGS*, we conduct a set of document clustering experiments on various datasets, and compare the performance of *GPGS* with a set of state-of-the-art methods. These experiments are discussed in Chapter 5.

Chapter 5

HARD DOCUMENT CLUSTERING

5.1 Introduction

Hard document clustering is a fundamental problem in *NLP* and *IR*, and is defined as:

Hard Document Clustering (HDC):

Given:

- A set $\mathbb{D} = \{D_1, \ldots, D_n\}$ of documents.

Seek:

A partitioning of $\mathbb{D}$[1], where the documents in each partition are more semantically similar to the ones in the same partition than to others.

In this chapter, a set of document clustering methods, based on *DSCTP*, *APV*, *GPGS* and a collection of state-of-the-art document representations: *NASARI*, *Doc2Vec*, *Sent2Vec*, *LexVec*, *fastText* and *GloVe* (reviewed in Section 1.2) respectively, are used to solve the *HDC* problem. The objectives of these experiments include:

(1) Further evaluate the effectiveness of *DSCTP*[2];

(2) Evaluate the effectiveness of *APV* and *GPGS*;

[1] To be clear, the partitioning of $\mathbb{D}$ is disjoint.

[2] Recall that a set of *document semantics comparison* experiments are presented in Section 2.5.

These experiments are presented in this chapter as three document clustering experiments over two different datasets. There are four major aspects to the experiments: **datasets**, **document clustering methods**, **performance evaluation methods**, and **experimental results**. We provide a brief overview for each of them below, and detail them in the remainder of this chapter.

Datasets:

To evaluate the performance of the document clustering methods based on *DSCTP*, *APV* and *GPGS* respectively, three datasets, *20Newsgroups* [78], *Reuters-21578* [84] and *BBC* [53], are in use. Each document in the datasets is assigned one or multiple semantic labels (e.g. "*ship*", a label for *shipping affairs*, in *Reuters-21578*) by human judges. Because of some shortcomings of *20Newsgroups* and *Reuters-21578* (e.g. duplicate text and confused categories) when they are directly used for hard document clustering, and also because of some empirical concerns (e.g. the limitation of our computation resources), we sampled subsets of documents from *20Newsgroups*, *Reuters-21578* and *BBC*. The samplings are discussed in Section 5.2.

Document Clustering Methods:

Recall that none of the document clustering methods have been detailed up to this point. *DSCTP*, *APV* and *GPGS* are not document clustering methods. Specifically, *DSCTP* is a document semantics comparison method, and *APV* and *GPGS* are document semantic representations. In order to perform document clustering, these methods have to be integrated with a clustering approach. In our experiments, we utilize spectral clustering as the clustering approach. On the other hand, *NASARI*, *Doc2Vec*, *Sent2Vec*, *LexVec*, *fastText* and *GloVe* also work with spectral clustering. Why spectral clustering is chosen and how *DSCTP*, and all considered *DSRs* will be integrated with spectral clustering to solve the *HDC* problem will be detailed in Section 5.3.

Performance Evaluation Methods:

Since the *HDC* problem is a special case of the general hard clustering problem, the quality measures for general hard clustering can also be used for document clusterings. Specifically, since the items in the datasets in our experiments are labeled (also called **ground truth classes**), the quality of document clusterings will be measured on how well the clusters, produced by a document clustering method, match the labels. We select three typical measures to evaluate the quality of clusterings: *adjusted rand index (ARI)* [64], *normalized mutual information (NMI)* [155], and *Fowlkes Mallows index(FMI)* [46]. These performance evaluation methods are discussed in Section 5.4.

Experimental Results:

The performances of those document clustering methods are compared by utilizing the evaluation methods mentioned above. In brief, the experimental results show that: first, the *DSCTP*-based method dominantly outperforms most of other methods; second, the *APV*-based and *GPGS*-based methods give competitive performance with *Doc2Vec*-based and *NASARI*-based methods; and third, the *GPGS*-based method provides improved performance to the *APV*-based method. Section 5.5 focuses on the experimental results and analysis.

5.2 Datasets

As mentioned earlier, three datasets are used to conduct our experiments: *20Newsgroups*, *Reuters-21578* and *BBC*. Because of some issues with the datasets and some empirical concerns, we sample *20Newsgroups*, *Reuters-21578* and *BBC*, thereby make two subsets from *20Newsgroups*, one subset from *Reuters-21578* and one subset from *BBC*. For convenience, we name these subsets *20News-M5*, *20News-C10*, *Reuters-M7* and *BBC-M5* respectively. Here we explain how and why we make *20News-M5*, *20News-C10*, *Reuters-M7* and *BBC-M5*.

20Newsgroups contains 18828 documents in 20 categories. These 20 categories are further organized in the hierarchical topic structure shown in Figure 5.1. However, there are several issues if it is directly used in our experiments.

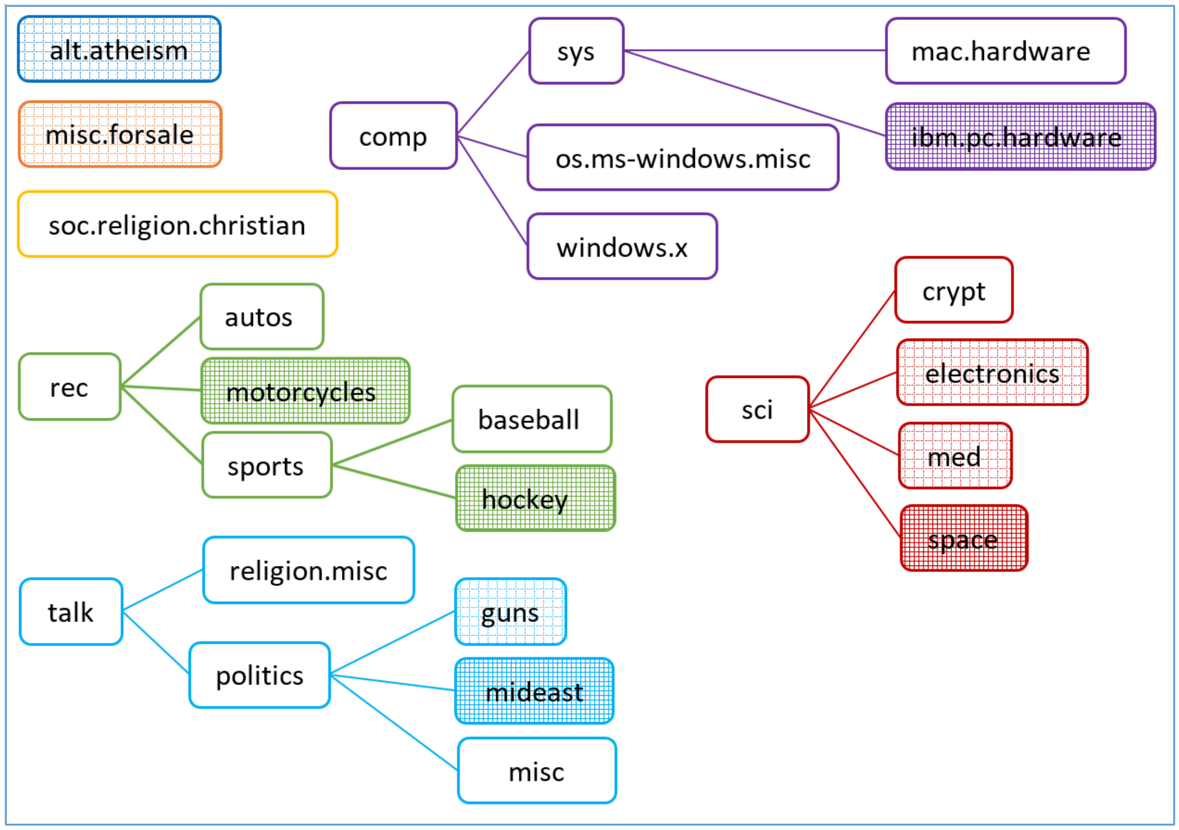
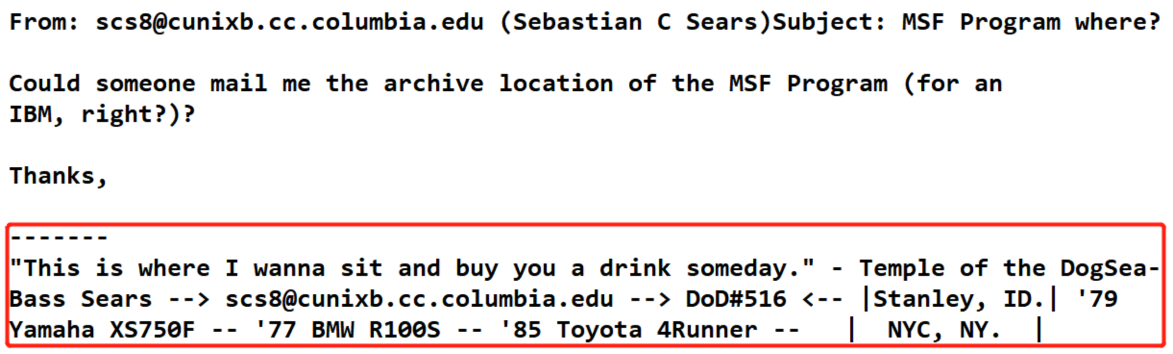


Figure 5.1: Category hierarchy of *20Newsgroups*. At the top level, there are 7 categories: *alt.atheism*, *misc.forsale*, *soc.religion.christian*, *rec*, *talk*, *comp* and *sci*. Shadowed boxes (including both light shadow and dark shadow) are the selected categories for *20News-C10*, in which dark-shadowed boxes are the categories for *20News-M5*.

Issue #1: Many categories are confused because of semantic overlap. For example, documents in the following categories *"soc.religion.christian"*, *"talk.politics.mideast"*, *"talk.religion.misc"* and *"alt.atheism"* share a large amount of similar semantics. Confused categories can form more challenging document clustering tasks; however, too much semantic overlap will not be helpful in evaluating the performance of document clustering methods.

Issue #2: Many documents contain tables and long signatures which do not contribute to the document semantics. For example, in the document *"rec.sports.hockey/53543"*, the primary content is merely a list of scores. Although this document may be meaningful in the *topic modeling* problem[3], it could hardly be meaningful in the *HDC* problem, because there are nearly no sentences to carry semantics. Signatures, on the other hand, should not be taken as a part of a document, because they are irrelevant to the document semantics. Thus, long signatures, sometimes containing sentences, would introduce a non-trivial amount of noise to document semantics. Such an example is shown in Figure 5.2.



```
From: scs8@cunixb.cc.columbia.edu (Sebastian C Sears)Subject: MSF Program where?

Could someone mail me the archive location of the MSF Program (for an
IBM, right?)?

Thanks,

-------
"This is where I wanna sit and buy you a drink someday." - Temple of the DogSea-
Bass Sears --> scs8@cunixb.cc.columbia.edu --> DoD#516 <-- |Stanley, ID.| '79
Yamaha XS750F -- '77 BMW R100S -- '85 Toyota 4Runner --   |  NYC, NY.  |
```

Figure 5.2: The content of *"rec.motorcycles/104551"*. The red box shows the customized signature. This document contains in total 420 characters, and the signature occupies 239 from them, and it does contain a sentence.

Issue #3 The size of *20Newsgroups* is considerable, and there are many large documents (e.g. *"alt.atheism/53519"* is 50KB). Since our computation resources are limited, processing the entire *20Newsgroups* dataset would be difficult.

Because of the three issues, we sample *20Newsgroups* as follows:

Step 1: We select five categories with minimal overlap. They are: *"rec.motorcycles"*, *"rec.sports.hockey"*, *"talk.politics.mideast"*, *"comp.sys.ibm.pc.hardware"* and *"sci.space"*.

Step 2: Based on the five categories selected in **Step 1**, we select another five categories that have some semantic overlap with the five selected ones. They are:

[3] *Topic modeling* concentrates on words more than sentences or documents because conventionally topics are represented by a set of words.

"alt.athesim", *"misc.forsale"*, *"sci.electronics"*, *"sci.med"* and *"politics.guns"*. Note that, *"alt.athesim"* and *"talk.politics.mideast"* have overlap; *"misc.forsale"*, *"rec.motorcycles"*, *"comp.sys.ibm.pc.hardware"* and *"sci.electronics"* have overlap with each other; *"politics.guns"* and *"talk.politics.mideast"* have overlap; and *"sci.med"*, *"sci.space"*, *"sci.electronics"* have overlap with each other.

Step 3: For each of the ten categories selected in **Step 1** and **Step 2**, we select 50[4] documents each of size of approximately $1KB$ and each contains no tables or long signatures. The resulting dataset covering the five categories selected in **Step 1** is named **20News-M5**, and the resulting dataset covering the ten categories selected in both **Step 1** and **Step 2** is named **20News-C10** shown in Figure 5.1. The IDs of all selected documents are listed in Appendix B.1.

Reuters-21578 has a different organization from *20Newsgroups*. First, not all documents in the dataset have category labels, and second, a document can have multiple categories. There are 10377 documents that belong to at least one category, and there are 119 categories. Among these categories, 36 of them have more than 50 documents[5]. However, a large portion of these 36 categories share many documents (i.e. documents assigned multiple labels). For example, *"soybean"* and *"corn"* have 111 and 224 documents in total respectively, but they have 48 shared documents. *Reuters-21578* also has the same issue as **Issue #2**. For example, the following two categories: *"money-fx"* and *"money-supply"*; and the following five categories: *"wheat"*, *"corn"*, *"soybean"*, *"grain"* and *"oilseed"*. In addition, *Reuters-21578* has many "near-duplicate" documents. For example, consider the following documents in *Reuters-21578*:

There were ***seven*** *grain ships loading and* ***six*** *ships were waiting to load at Portland, according to the Portland Merchants Exchange.*

"ship/106"

[4] Accoring to [132], 50 is a rule of thumb number for a dataset.

[5] These 36 categories are listed in Appendix B.2.2

*There were **three** grain ships loading and **two** ships were waiting to load at Portland, according to the Portland Merchants Exchange.*

"ship/3386"

*There were **five** grain ships loading and **three** ships were waiting to load at Portland, according to the Portland Merchants Exchange.*

"ship/12327"

*There were **six** grain ships loading and **six** ships were waiting to load at Portland, according to the Portland Merchants Exchange.*

"ship/14688"

Among these four documents, the only differences are two numbers in each sentence at the same positions. This issue of "near-duplicate" documents has been studied in some previous work such as [127] and [116]. We note that these "near-duplicate" documents are not helpful in evaluating the performance of document clustering methods. Finally, **Issue #3** in *20Newsgroups* is also an issue in *Reuters-21578*. Thus, taking all concerns above into consideration, we select 7 categories: *"interest"*, *"ship"*, *"livestock"*, *"soybean"*, *"gold"*, *"crude"* and *"acq"*. Similar to *20News-M5* and *20News-C10*, 50 documents are selected from each of the 7 categories. The selected documents are each of size of approximately $1KB$ and have only one label in the 7 categories. Additionally any "near-duplicate" documents will be excluded. We name the resulting dataset **Reuters-M7**.

BBC [53] contains 2225 documents in 5 categories. We sample 50 documents for each category, and each document is of size of approximately $1KB$. Similar to *Reuters*, *BBC* also contains "near-duplicate" documents. We exclude any "near-duplicate" documents when sampling.

1

5.3 Document Clustering Methods

As mentioned earlier, since *DSCTP*, *APV*, *GPGS*, *Doc2Vec*, *NASARI*, *Sent2Vec*, *LexVec*, *fastText* and *GloVe* are not document clustering methods, we integrate them with *spectral clustering* to address the *HDC* problem. In this section, we firstly explain why we choose spectral clustering, and secondly explain how these methods are integrated with spectral clustering.

Recall that, *DSCTP* is a document semantics comparison method that takes two documents as inputs and produces a similarity value. *APV*, *GPGS*, *NASARI*, *Doc2Vec*, *Sent2Vec*, *LexVec*, *fastText* and *GloVe* are *DSRs*. For these latter methods, a similarity value can be computed simply by using *cosine similarity* based on the specific representations. Thus, the clustering method to be chosen should be distance-based[6]. There are three major categories of distance-based clustering methods: *k-means-based* methods, *hierarchical* methods and *space-mapping-based* methods. We use spectral clustering, one of the *space-mapping-based* methods. Here are the reasons. There are two issues with *k-means-based* methods. One is that those methods assume that the clusters are convex (i.e. roughly speaking a cluster is of a "ball" shape). However, this assumption does not necessarily hold in the *HDC* problem. The other issue is that *k-means-based* methods require *Euclidean distance* (or other distance functions)[7] to be defined in the space of data points. However, there is no valid distance function known to us defined upon *DSCTP* or the four document semantic representations because the *triangle inequality* property required by distance functions is not necessarily satisfied. More specifically, in terms of document semantics, if the distance between two documents A and B is $d(A, B)$, and the distance between A and another document C is $d(A, C)$, then it may not be true that $d(A, B) + d(A, C) \geq d(B, C)$ (i.e. B and C can be totally dissimilar, and their similarity can be zero but their distance can be infinity). *Hierarchical* methods (e.g. *agglomerative clustering*) also have two issues.

[6] Similarities can be converted to distances (e.g. $d = -\ln(s)$)

[7] Formally, a *distance function* is required to satisfy four properties: first, non-negativity; second, zero distance to self; third, symmetry; and fourth, triangle inequality.

One is the same distance function issue as that in *k-means-based* methods. The other is that *hierarchical* methods additionally rely on *linkage criteria* [101] to merge lower level clusters to form higher level clusters hierarchically. Although indeed there exist a collection of known linkage criteria, determining the best linkage criteria for our problem is decidedly non-trivial and considered as a future work. On the other hand, *spectral clustering*, as a typical *space-mapping-based* method, only requires pairwise similarities between data points, and thus can naturally be integrated with *DSCTP* and the four document semantic representations (with the cosine similarity).

To integrate with *spectral clustering*, we need to compute pairwise document semantic similarities by utilizing *DSCTP*, and the involved *DSRs*. Here we detail how *DSCTP* and the *DSRs* are thus utilized. Note that *Doc2Vec*, *NASARI*, *Sent2Vec*, *LexVec*, *fastText* and *GloVe* require text preprocessing. Tokenization is required by all of them. *LexVec* and *fastText* require the removal of all punctuations, lowercase text, and the conversions of numbers to words. We also remove stopwords before using these models. Next, we detail the settings for these methods.

DSCTP: *DSCTP* takes two documents and a word similarity threshold θ_w as inputs, and outputs a real-valued similarity. In our experiments, setting $\theta_w = 0.3$ leads to the best experimental results.

APV & GPGS: Both *APV* and *GPGS* require a background document set and a *GP* similarity threshold θ_{GP} to proceed. For the two experiments on *20Newsgroups*, we use *20News-C10* as the background document set; and for the experiment on *Reuters-21578*, we use *Reuters-M7* as the background document set. We set $\theta_{GP} = \theta_w = 0.3$ for both *APV* and *GPGS*. In addition, *APV* requires a graph diameter threshold θ_{dia}, and it is set to 3 in our experiments. Recall that *APV* can be constructed by two methods, the *PRSC*-based *APV-GPC*[8] and the *BRSC*-based *APV-GPC*. Both of these two methods are tested in our experiments. *GPGS* requires a parameter h for h-NNGs and a finite set T of

[8] Recall that *APV-GPC* is the name of our *GP*-clustering-based *APV* construction method.

tau's for the Heat kernel bank. We set $h = 1$ and $T = \{1, 5, 10\}$ to obtain the best experimental results. To compute the pairwise document semantic similarity, we use the cosine similarity.

Other *DSRs*: These *DSRs* have been reviewed in Section 1.2. We use pre-trained models of these *DSRs* as provided by their developers. For *NASARI*, we use their word vector model (300 dimensions) trained on the *UMBC* corpus [56][9]. For *Dov2Vec*, we use their model (300 dimensions) trained on English *Wikipedia*[10]. For *fastText*, we use their 1 million word vector model (300 dimensions) trained on *Wikipedia 2017* and the *UMBC* corpus[11]. For *Sent2Vec*, we use their *unigrams* model (700 dimensions) trained on English tweets[12]. For *LexVec*, we use their word vector model (300 dimensions) trained on the *Common Crawl* corpus [34][13]. For *GloVe*, we use their word vector model (300 dimensions) trained on *Wikipedia 2014* and the *Gigaword 5* corpus[14]. Among these models, *Doc2Vec* can directly take a document that contains multiple sentences as input. *Sent2Vec* requires an input document be split into a collection of sentences, and takes this collection as input. The Other *DSRs* are all word vector models. Thus, we need to, first, tokenize an input document into a collection of tokens; second, send each token into a *DSR* model and obtain a vector for that token; and finally, sum (element-wise sum) all obtained vectors to get a single vector for the input document[15]. By using these pre-trained models, the semantic similarity between

[9] *NASARI* pre-trained models can be downloaded from: http://lcl.uniroma1.it/nasari/.

[10] *Doc2Vec* pre-trained models can be downloaded from: https://github.com/jhlau/doc2vec/.

[11] *fastText* pre-trained models can be downloaded from: https://fastText.cc/.

[12] *Sent2Vec* pre-trained models can be downloaded from: https://github.com/epfml/sent2vec/.

[13] *LexVec* pre-trained models can be downloaded from: https://github.com/alexandres/lexvec/.

[14] *GloVe* pre-trained models can be downloaded from: https://nlp.stanford.edu/projects/glove/.

[15] When applying word vector models to documents, averaging (element-wise) the word vectors of the words in the document is another approach to obtain a single vector to represent the document. Nonetheless, since we will use cosine similarity to compute the similarity value between two documents,

two documents can be directly computed by cosine similarity.

As mentioned above, to compute document similarities over the *DSRs*, we use *cosine similarity*. Note that other similarity or distance approaches may also be applicable. For example, Euclidean distance will be applicable when the vectors are normalized. It is straightforward to verify that Euclidean distance is determined by the square root of cosine similarity in the normalized vector case, because the vector lengths are all 1. Cosine similarity and Euclidean distance on normalized vectors utilize angles between vectors instead of lengths to compute similarities and distances. On the other hand, some studies show that, in the context of vector-formed semantic representations such as *Word2vec* [97], lengths of vectors are also meaningful and reflect additional distance information of vectors [83] [131]. Additionally, the family of *Bregman distance* [17] may also applicable when some conditions are satisfied. For example, *Kullback-Leibler divergence* [76] is one of the most celebrated members of *Bregman distance*. Its symmetric variant *Jensen-Shannon divergence* [32] can be used to compute distances based on the vector representations if the vector values can be appropriately converted to probabilities[16]. We leave this topic to future work.

When applying spectral clustering, we use *normalized Laplacian matrix* to balance the sizes of clusters [157], and also we keep all eigenvectors so as to perform the space mapping. Spectral clustering also takes the desired number of clusters as an input. In our experiments, for each dataset (e.g. *20News-M5*), it would be natural to assume that the number of categories is the perfect desired number of clusters. Nonetheless, it can happen that a document clustering method may not obtain its best performance with the initial number of clusters, and if we let it keep partitioning the dataset, then better performance may be achieved. The reason is that a clustering

sum and average, in this case, have no impact on the result. In addition, since *Word2Vec* explicitly proposes the algebraic operation properties (mentioned in Chapter 1), then to keep consistent with these properties, we use sum instead of average.

[16] This conversion needs to be reasonable in both of its mathematical form and the corresponding meaning with respect to document semantics.

method may be weak or sensitive in discriminating some particular documents, and thus it may output noised clusters. On the other hand, the three measures of quality of clusterings (i.e. *ARI*, *NMI* and *FMI*) will not keep producing better results when a clustering method keeps partitioning the dataset. Hence, for each dataset, if it contains K categories, we set the desired number of clusters for spectral clustering to K, $K+1$, $\ldots$, $2K$ respectively, and in this work we only consider the performance of a document clustering method within this range.

For convenience, in the remainder of this chapter, we denote the document clustering method using *DSCTP* as **DSCTP**; denote the method using *TRSC*-based *APV* as **APV-TRSC**; denote the method using *BRSC*-based *APV* as **APV-BRSC**; denote the method using *GPGS* as **GPGS**; denote the method using *Doc2Vec* as **Doc2Vec**; denote the method using *NASARI* as **NASARI**; denote the method using *Sent2Vec* as **Sent2Vec**; denote the method using *LexVec* as **LexVec**; denote the method using *fastText* as **fastText**; and denote the method using *GloVe* as **GloVe**.

5.4 Evaluation Methods

Each of the document clustering methods proposed in Section 5.3 will be tested on the four datasets proposed in Section 5.2. As discussed in Section 5.1, the performance of the document clustering methods will be evaluated by measuring the quality of clusterings produced by these methods. In our experiments, the measures for the quality of clusterings include *ARI* [64], *NMI* [155] and *FMI* [46]. Intuitively, each of these three measures evaluates how well two clusterings agree with each other. More specifically, in our experiments, we use these three methods to evaluate how well the clusterings produced by the document clustering methods agree with the ground truth classes. We choose these three measures because they evaluate the relatedness between two clusterings from different perspectives. *ARI* is a statistical measure based on the number of agreements, while *FMI* is also a statistical measure yet based on *recall* and *precision*. Different from *ARI* and *FMI*, *NMI* measures the dependence between two clusterings from the information theory point of view (i.e. it measures how much information of a clustering can be obtained when the other clustering is given). Here we briefly review these measures.

ARI measures the similarity between two clusterings. ARI is an improved version of the *rand index (RI)*. RI works as follows: Let $X = \{x_1, \ldots, x_N\}$ denote a set of N data points, let $\mathbb{A} = \{A_1, \ldots, A_M\}$ denote a clustering of X, and let $\mathbb{B} = \{B_1, \ldots, B_K\}$ denote another clustering of X. Then $\binom{N}{2}$ is the number of all possible unordered pairs of data points from X. Let s denote the number of all unordered pairs of data points that are in the same cluster in both $\mathbb{A}$ and $\mathbb{B}$. For example, suppose $\{x_i, x_j\}$, where $x_i, x_j \in X$, is a pair of data points. If $x_i, x_j \in A_h$ and $x_i, x_j \in B_l$, then $\{x_i, x_j\}$ contributes 1 for s. Let d denote the number of unordered pairs of data points that are not in the same cluster in $\mathbb{A}$ nor in the same cluster in B. Then,

$$RI = \frac{s+d}{\binom{N}{2}} \tag{5.1}$$

RI takes values ranging from 0 to 1. Note that 0 means that A and B are totally dissimilar; and 1 means that they are totally similar. However, RI has an issue. That is, when evaluating the quality of clusterings, the baseline quality should come from a random clustering, but the minimum possible value of RI (i.e. 0) does not necessarily correspond to a random clustering. To correct this, ARI was introduced in [64], and defined as: Let the *contingency table* of A and B be denoted by $[n_{hl}]$, where n_{hl} is the number of data points clustered to A_h in $\mathbb{A}$ and clustered to B_l in $\mathbb{B}$. Then from [64]

$$ARI = \frac{RI - \text{Expected } RI}{\text{Max } RI - \text{Expected } RI} = \frac{\sum\limits_{h,l} \binom{n_{hl}}{2} - \frac{\sum\limits_{h} \binom{\sum_l n_{hl}}{2} \sum\limits_{l} \binom{\sum_h n_{hl}}{2}}{\binom{N}{2}}}{\frac{1}{2}\left[\sum\limits_{h} \binom{\sum_l n_{hl}}{2} + \sum\limits_{l} \binom{\sum_h n_{hl}}{2}\right] - \frac{\sum\limits_{h} \binom{\sum_l n_{hl}}{2} \sum\limits_{l} \binom{\sum_h n_{hl}}{2}}{\binom{N}{2}}} \tag{5.2}$$

ARI takes values ranging from -1 to 1 with 1 indicating that $\mathbb{A}$ and $\mathbb{B}$ totally match, -1 indicating that $\mathbb{A}$ and $\mathbb{B}$ are independent, and 0 indicating that $\mathbb{A}$ and $\mathbb{B}$ are random relative to each other.

FMI is a measure based on *recall* and *precision*. We use the notations above. *FMI* [46] is computed by

$$FMI = \sqrt{\text{recall} \cdot \text{precision}} = \sqrt{\frac{TP}{TP + FN} \cdot \frac{TP}{TP + FP}} \tag{5.3}$$

where $TP = s$, FP denotes the number of unordered pairs of data points that are in the same cluster in $\mathbb{A}$ but not in the same cluster in $\mathbb{B}$, and FN denotes the number of unordered pairs of data points that are not in the same cluster in $\mathbb{A}$ but are in the same cluster in $\mathbb{B}$. *FMI* takes values from 0 to 1 with 0 indicating that $\mathbb{A}$ and $\mathbb{B}$ are independent, and 1 indicating that $\mathbb{A}$ and $\mathbb{B}$ are equivalent.

NMI [155] is the normalized version of *mutual information (MI)*. *MI* is defined as follows: The probability of a data point in X being clustered to A_h in $\mathbb{A}$ is computed by $P_{A_h} = \frac{|A_h|}{N}$, and the probability of a data point being clustered to A_h in $\mathbb{A}$ and being clustered to B_l in $\mathbb{B}$ is computed by $P_{A_h B_l} = \frac{|A_h \cap B_l|}{N}$. Then, *MI* [32] of A and B is computed by

$$MI = \sum_{h=1}^{M} \sum_{l=1}^{K} P_{A_h B_l} \log \left(\frac{P_{A_h B_l}}{P_{A_h} P_{B_l}} \right) \tag{5.4}$$

Let $H(\mathbb{A}) = -\sum_{h=1}^{M} P_{A_h} \log(P_{A_h})$ denote the *entropy* of $\mathbb{A}$, and let $H(\mathbb{B})$ denote the *entropy* of $\mathbb{B}$. Then *NMI* is computed by[17]

$$NMI = \frac{MI}{\frac{H(\mathbb{A})+H(\mathbb{B})}{2}} = \frac{2}{\frac{H(\mathbb{A})}{MI} + \frac{H(\mathbb{B})}{MI}} \tag{5.5}$$

NMI takes values from 0 to 1 with 0 indicating independent clusterings, and 1 indicating equivalent clusterings.

[17] This formula shows that *NMI* is also the *symmetric uncertainty coefficient* [155].

5.5 Experimental Results

In this section, we firstly profile *APV-BRSC*, *APV-TRSC* and *GPGS* by providing a set of statistics (e.g. vector length per document). Details are given in Section 5.5.1.

Secondly, we give the experimental results of utilizing the 10 document clustering methods (i.e. methods respectively based on *DSCTP*, *APV-BRSC*, *APV-TRSC*, *GPGS*, *Doc2Vec*, *NASARI*, *Sent2Vec*, *LexVec*, *fastText* and *GloVe*) to cluster documents in the four datasets (i.e. *20News-M5*, *20News-C10*, *Reuters-M7* and *BBC-M5*) respectively. The results are shown in Tables 5.0, 5.1, 5.2, 5.3, 5.4 and 5.5, 5.6 and 5.7. In these tables, K is the number of clusters. *ARI*, *FMI* and *NMI* are used to measure the quality of clusterings. From the results, first, *DSCTP* significantly outperforms most of others; and second, *APV-BRSC*, *APV-TRSC* and *GPGS* provide competitive, and in some cases better, performance than the state-of-the-art methods. We detail the results in Section 5.5.2.

Finally, from the results we observed that *DSCTP* provides a superior performance. However, *APV-BRSC*, *APV-TRSC* and *GPGS* do not perform as well as *DSCTP* even though they are based on *DSCTP*. In section 5.6, we discuss this situation.

5.5.1 Profiles of APV and GPGS

To understand our three *DSRs*, *APV-BRSC*, *APV-TRSC* and *GPGS*, from empirical perspectives, we provide a collection of statistics in this section. Table 5.1 shows summary profiles of *APV-BRSC*, *APV-TRSC* and *GPGS* when they are applied to *20News-C10*[18], *Reuters-M7* and *BBC-M5*. From Table 5.1 several observations deserve to be highlighted. First, when the average number of non-trivial words of a document is less than 100, the average non-zero dimensions of *APV-BRSC* and *APV-TRSC* is typically less than 150. Thus, when computing cosine similarity between two

[18] Since *20News-M5* is a subset of *20News-C10*, we do not provide another individual statistic profile for *20News-M5*.

Statistic Profiles of APV and GPGS			
	20News-C10	**Reuters-M7**	**BBC-M5**
Documents	500	350	250
Total Words	7672	4758	6201
Total GPs	71640	87508	76989
BRSC APs	17194	18793	17462
TRSC Aps	16978	19089	17504
Avg. Words per Doc.	60	94	98
Avg. GPs per Doc.	5110	18978	10493
Avg. APV-BRSC	46	143	143
Avg. APV-TRSC	52	143	143
Avg. GPGS	1617	5190	3667

Table 5.1: **Documents** denotes the number of documents in each dataset. **Total Words** denotes the total number of non-trivial words (i.e. stopwords and punctuations are removed). **Total GPs** denotes the total number of *GPs* extracted from the documents in each dataset. **BRSC APs** and **TRSC APs** denote the numbers of *APs* (i.e. *GP* clusters) computed by *BRSC* and *TRSC* respectively for each dataset. **Avg. Words per Doc.** denotes the average number of non-trivial words contained in a document. **Avg. GPs per Doc.** denotes the average number of *GPs* contained in a document. **Avg. APV-BRSC**, **Avg. APV-TRSC** and **Avg. GPGS** denote the average numbers of non-zero dimensions in an *APV-BRSC*, in an *APV-TRSC* and in a *GPGS* respectively.

APV-BRSC's (or *APV-TRSC*'s), the required average number of common dimensions is less than 300, which coincides with the most commonly seen number of dimensions set by pre-trained *DSR* or word embedding models. Second, the average numbers of non-zero dimensions of *APV-BRSC* and *APV-TRSC* are not sensitive to the total number of *GPs* and the number of *APs* (regardless of *BRSC* or *TRSC*). Third, *GPGS* is quite different from *APV-BRSC* and *APV-TRSC* as the average number of non-zero dimensions of *GPGS*'s may vary significantly when the total number of *GPs* varies. Also, *GPGS*'s requires more dimensions than *APV-BRSC* and *APV-TRSC* to represent document semantics.

Figures 5.3, 5.4 and 5.5 visualize the relationship between the number of non-trivial words in a document and the number of *GPs* extracted from a document, the relationship between the relationship between the number of non-trivial words in a document and the non-zero dimensions of *APV-BRSC* and *APV-TRSC* respectively, and the relationship between the relationship between the number of non-trivial words in a document and the non-zero dimensions of *GPGS* with respect to *20News-C10*. Similarly, such relationships in *Reuters-M7* are visualized in Figures 5.6, 5.7 and 5.8; also those in *BBC-M5* are visualized in Figures 5.9, 5.10 and 5.11. It is straightforward to observe that in these relationships *Reuters-M7* and *BBC-M5* follow very similar patterns while *20News-C10* is different from those two. Specifically, the number of *GPs* and *DSR* dimensions have approximately linear relationships with the number of words in a document. On the other hand, *20News-C10* does not have such linear relationships in an explicit way. One possible reason is that the documents in *Reuters-M7* and *BBC-M5* are of news writings which are typically well written while the writing in *20News-C10* is relatively casual.

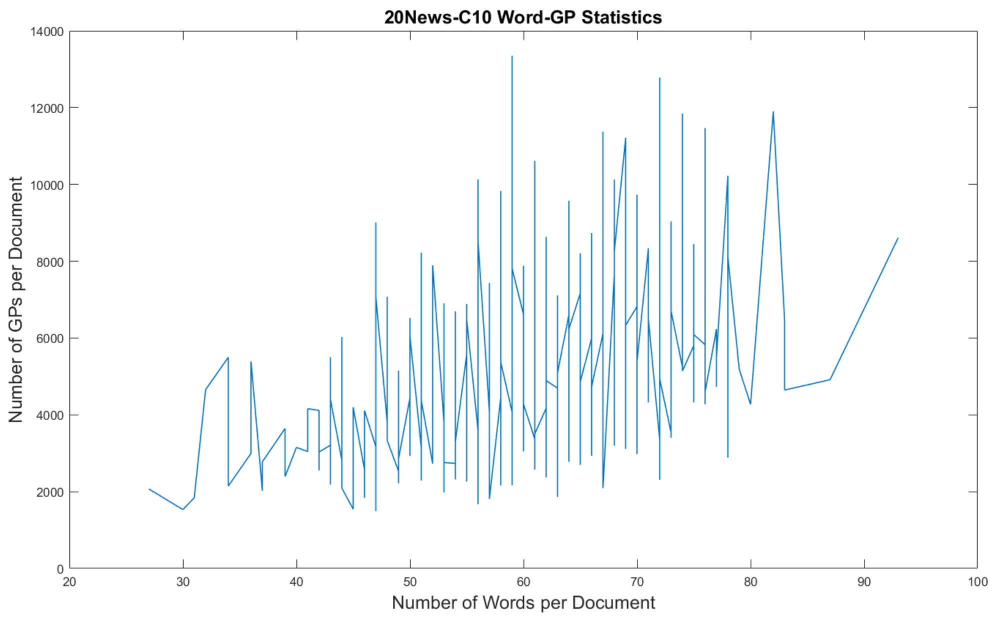


Figure 5.3: The relationship between the number of non-trivial words (i.e. non-stopwords) per document and the number of extracted *GPs* per document in *20News-C10*.

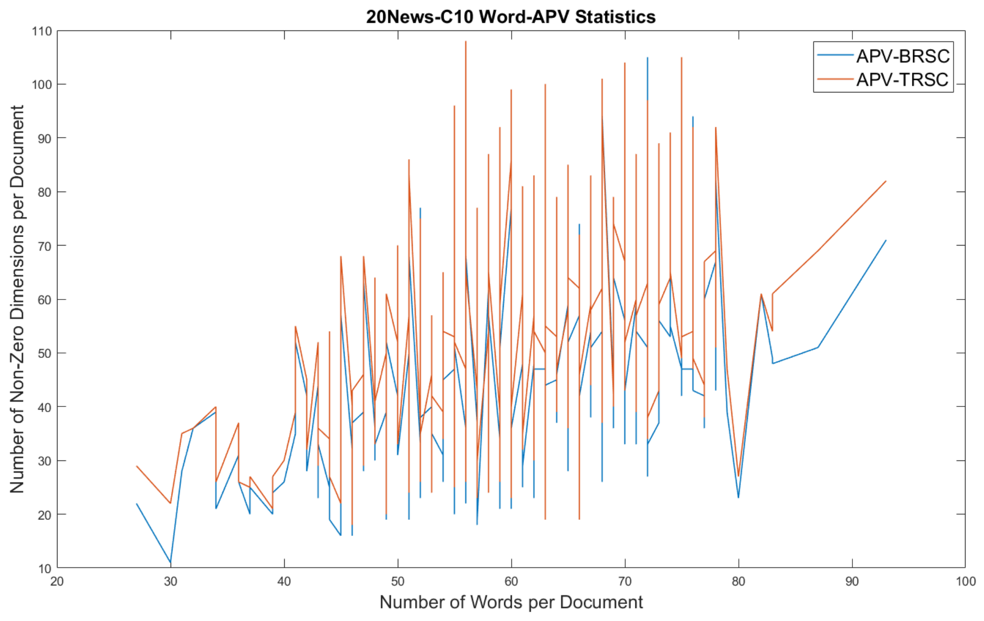


Figure 5.4: The relationship between the number of non-trivial words per document and the number of non-zero dimensions in an *APV* for a document in *20News-C10*. The blue line shows the relationship for *APV-BRSC*, and the orange line shows the relationship for *APV-TRSC*.

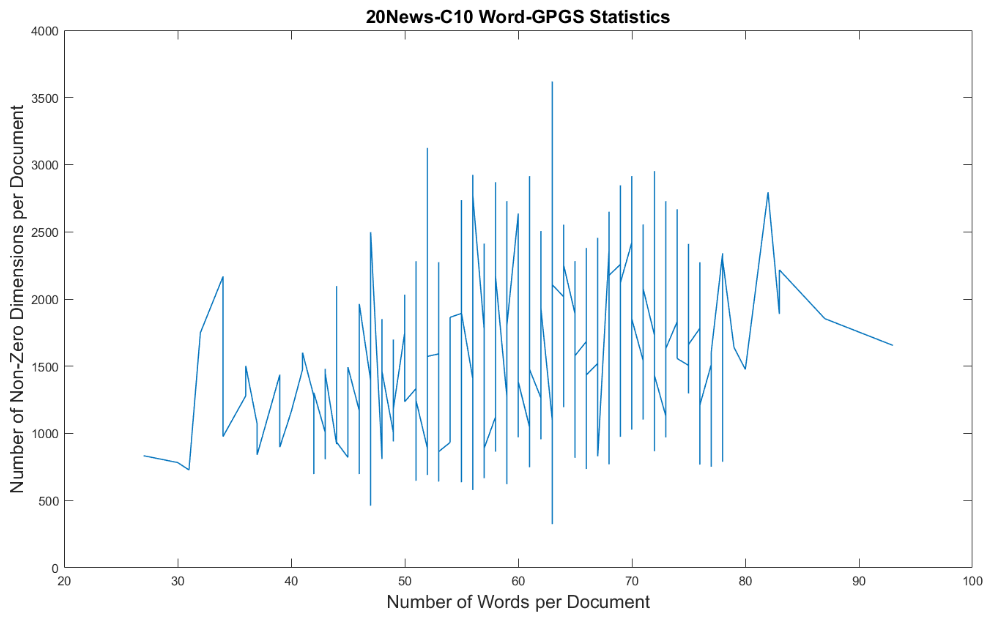


Figure 5.5: The relationship between the number of non-trivial words per document and the number of non-zero dimensions in a *GPGS* for a document in *20News-C10*.

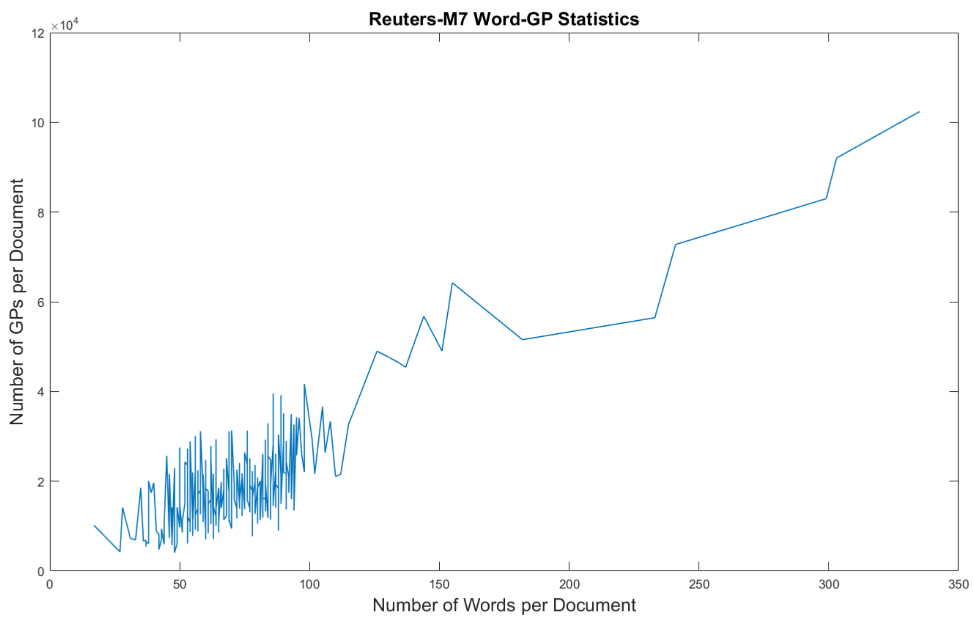


Figure 5.6: The relationship between the number of non-trivial words (i.e. non-stopwords) per document and the number of extracted *GPs* per document in *Reuters-M7*.

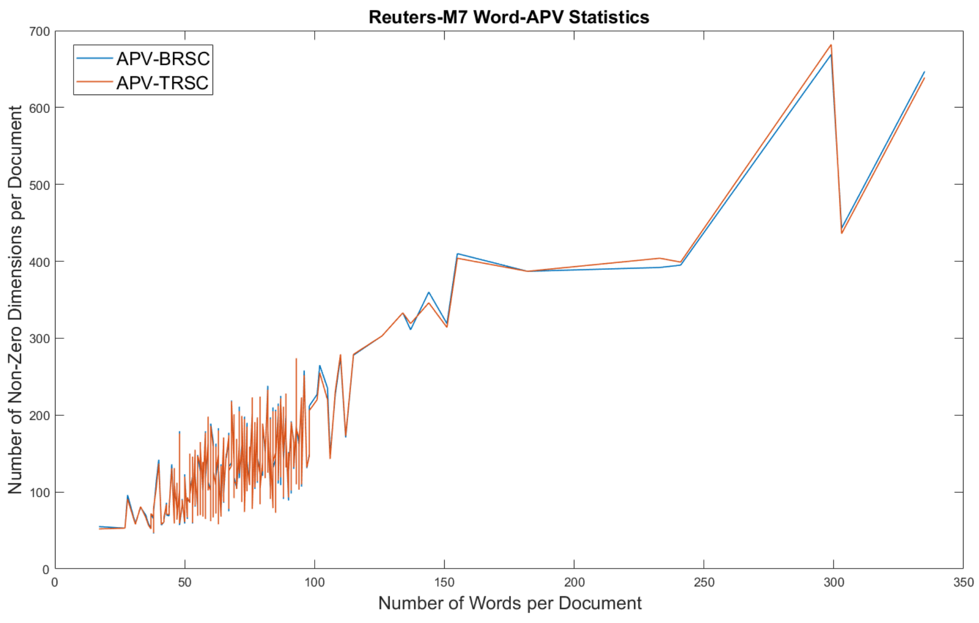


Figure 5.7: The relationship between the number of non-trivial words per document and the number of non-zero dimensions in an APV for a document in *Reuters-M7*.

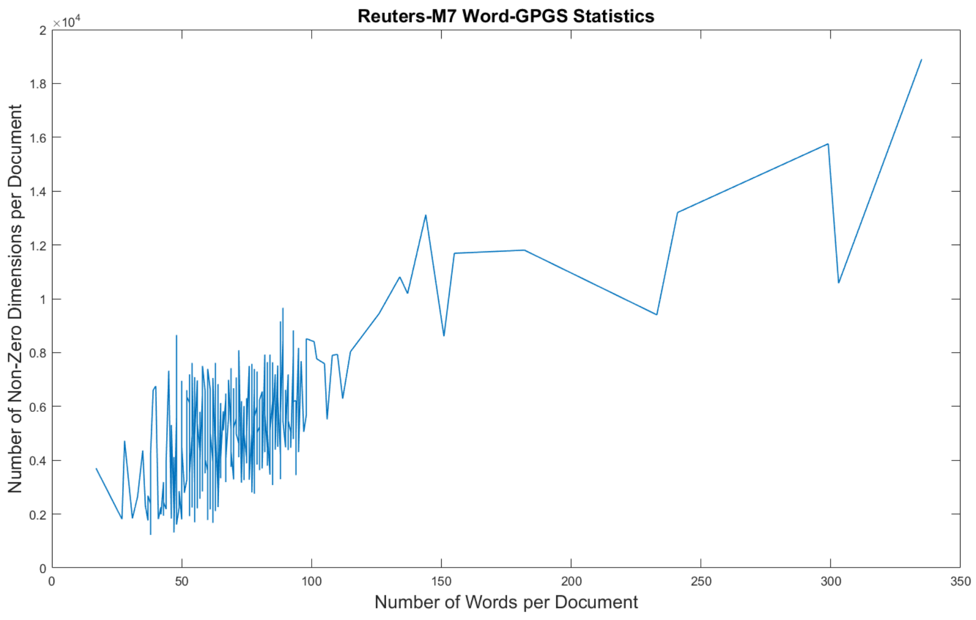


Figure 5.8: The relationship between the number of non-trivial words per document and the number of non-zero dimensions in a *GPGS* for a document in *Reuters-M7*.

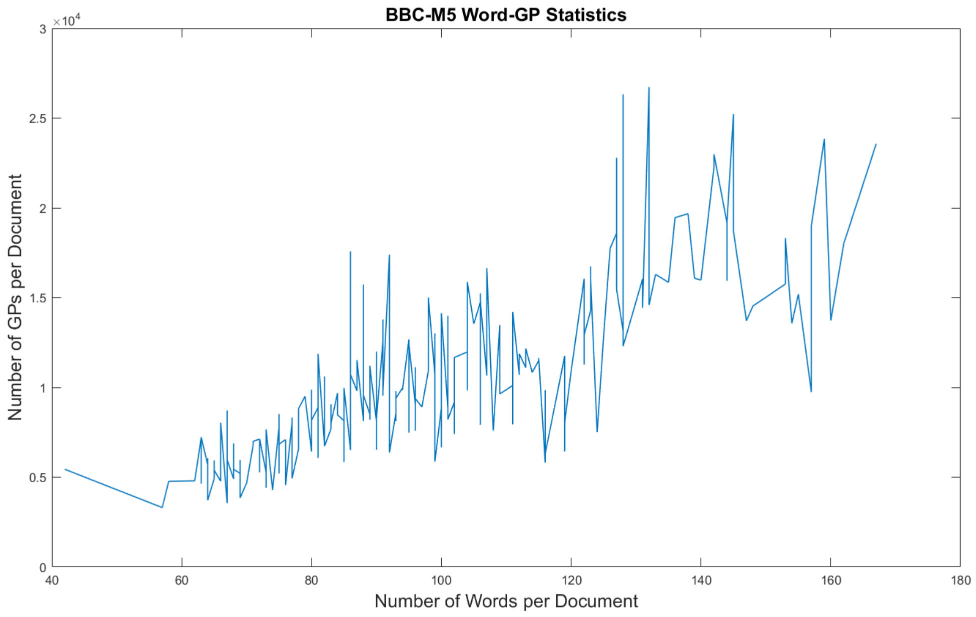


Figure 5.9: The relationship between the number of non-trivial words (i.e. non-stopwords) per document and the number of extracted *GPs* per document in *BBC-M5*.

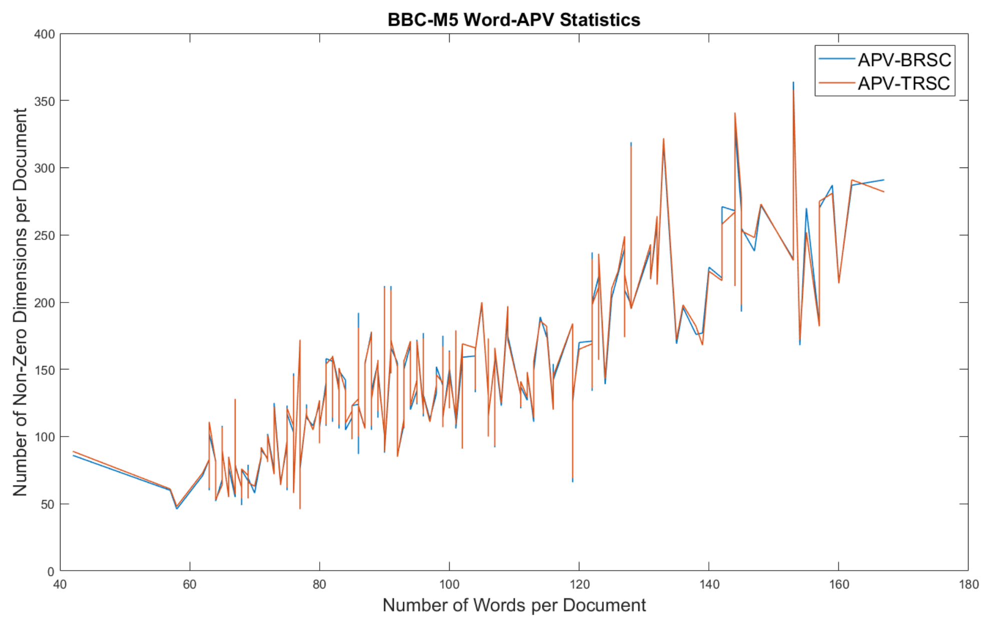


Figure 5.10: The relationship between the number of non-trivial words per document and the number of non-zero dimensions in an *APV* for a document in *BBC-M5*.

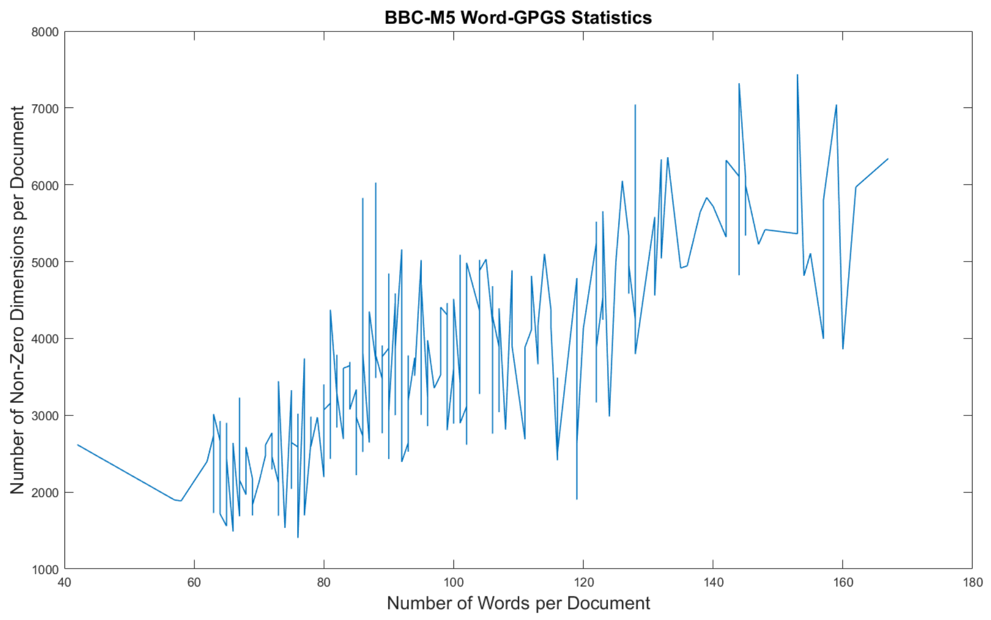


Figure 5.11: The relationship between the number of non-trivial words per document and the number of non-zero dimensions in a *GPGS* for a document in *BBC-M5*.

Figures 5.12 and 5.13 illustrate the empirical running times of constructing an *APV* and a *GPGS* respectively. The hardware environment is the same as that in Section 2.5. The running times are measured on *20News-C10*. The running time for the *APV* construction is measured from the start of **Algorithm 3.3** to the end of **Algorithm 3.4**. The running time for the *GPGS* construction is measured from the start of the ***Significance Value and Induced Value Assignment*** stage in **Algorithm 4.1** to the end of **Algorithm 4.1**. From these two figure, it is clear that the two running times are both linear to the number of distinct *GPs* extracted from a document. To construct one *APV*, the maximum running time in Figure 5.12 is 122 seconds, the minimum running time is 8 seconds, and the average running time is 23 seconds. To construct one *GPGS*, the maximum running time in Figure 5.13 is 163 seconds, the minimum running time is 12 seconds, and the average running time is 37 seconds. For the *APV* construction, since the *GP* graph construction and the *GP* clustering will not be run for each *APV* construction, we do not count them into the *APV* construction running time. The *GP* graph construction over 71640 *GPs* takes approximately 8 minutes. The *GP* clustering using *TRSC* over that *GP* graph takes approximately 40 hours, and the clustering using *BRSC* takes approximately 70 hours. For the *GPGS* construction, it also needs the *GP* graph construction, but it does not need the *GP* clustering.

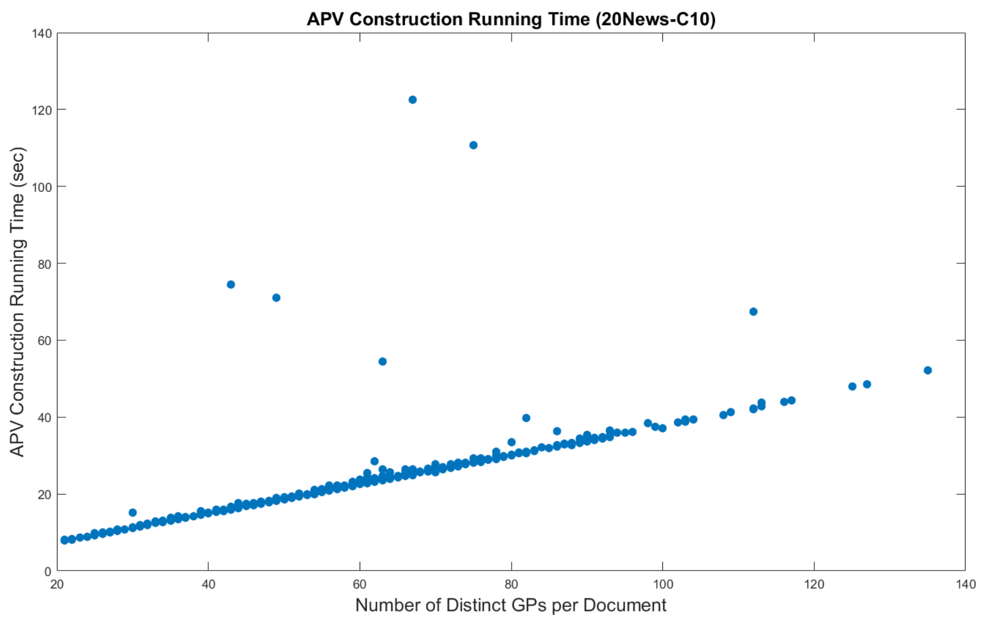


Figure 5.12: The relationship between the number of distinct *GPs* extracted from a document and the running time for constructing an *APV* based on the *GPs*.

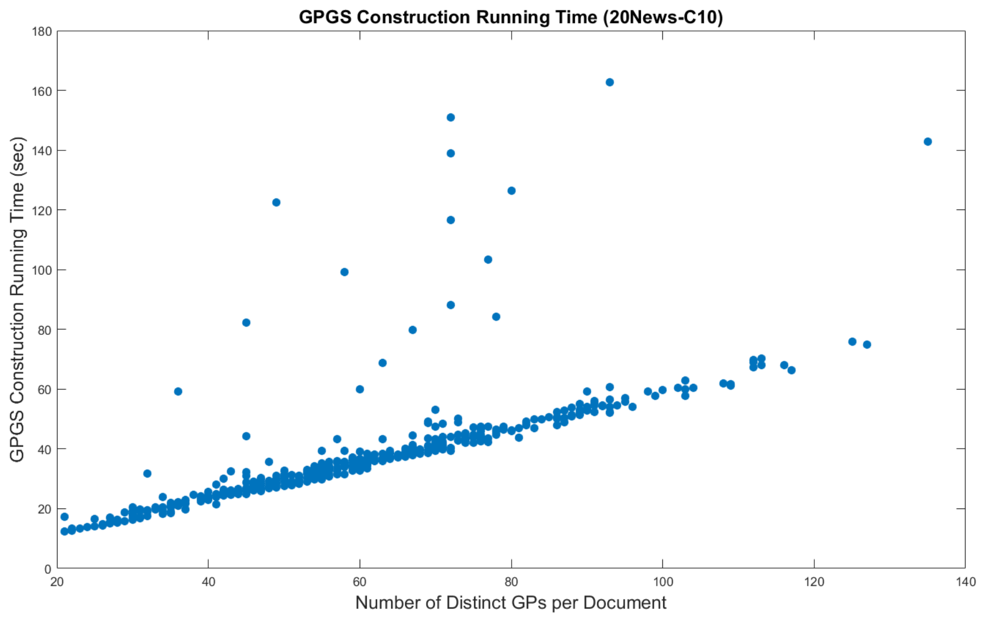


Figure 5.13: The relationship between the number of distinct *GPs* extracted from a document and the running time for constructing an *GPGS* based on the *GPs*.

The average running time of generating a representation vector for a single document by using *Doc2Vec* is 0.063 seconds. The average running time by using *NASARI* is 0.035 seconds. The average running time by using *Sent2Vec* is 0.004 seconds. The average running time by using *LexVec* is 0.002 seconds. The average running time by using *fastText* is 0.002 seconds. The average running time by using *GloVe* is 0.008 seconds. Similar to the running times of the state-of-the-art methods discussed in Chapter 2. These running times are based on pre-trained models. The training times and the model loading times are not counted, though they can be considerable.

5.5.2 HDC Results

Table 5.0 gives the results on *20News-M5*, and each subtable corresponds to a document clustering method. In each subtable, K takes values $5, 6, 7, 8, 9$ and 10, and the best *ARI*, *NMI* and *FMI* scores in that subtable are bold. To compare the document clustering methods, the *ARI*, *NMI* and *FMI* scores for each method at $K = 5$ are summarized in Table 5.1a, and the best scores for each method among all K's are summarized in Table 5.1b. These summarized results show that *DSCTP* is the top method for the *HDC* tasks on *20News-M5*, and that *APV-BRSC*, *APV-TRSC* and *GPGS* provide similar performance to *DSCTP*, while outperforming the other methods. In addition, *DSCTP* and our *DSRs* achieve their best scores at $K = 5$ while many others have to keep partitioning the dataset to reach their best scores. Additionally, *APV-TRSC* provides a better performance than *APV-BRSC*. This is an evidence to show that when constructing *APVs* *TRSC* can be a better option than *BRSC*. Besides, *GPGS* shows a better performance than *APV-BRSC* and *APV-TRSC*, which justifies that *GPGS* can provide better performance than *APV*.

Similarly, Tables 5.2 and 5.3 show the results for *20News-C10*. On this dataset, *DSCTP* leads at $K = 10$, though *LexVec* achieves the same *ARI* and *FMI* at $K = 11$ and a slightly better *NMI* at $K = 15$. Also, *APV-BRSC*, *APV-TRSC* and *GPGS* perform similarly to *DSCTP*, and outperform *Doc2Vec* and *Sent2Vec*. Compared with the performance of all methods on *20News-M5*, the methods receive a

20News-M5 DSCTP						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.90**	0.89	0.81	0.71	0.60	0.57
NMI	**0.91**	0.91	0.83	0.78	0.72	0.71
FMI	**0.92**	0.91	0.85	0.77	0.67	0.66

(a) *DSCTP*

20News-M5 APV-BRSC						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.84**	0.78	0.78	0.76	0.73	0.70
NMI	**0.86**	0.81	0.82	0.80	0.79	0.78
FMI	**0.87**	0.82	0.83	0.81	0.78	0.76

(b) *APV-BRSC*

20News-M5 APV-TRSC						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.87**	0.84	0.78	0.76	0.74	0.72
NMI	**0.87**	0.85	0.81	0.80	0.79	0.78
FMI	**0.9**	0.87	0.82	0.81	0.79	0.77

(c) *APV-TRSC*

20News-M5 GPGS						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.88**	0.83	0.78	0.74	0.72	0.71
NMI	**0.87**	0.84	0.82	0.79	0.79	0.77
FMI	**0.90**	0.86	0.83	0.79	0.78	0.77

(d) *GPGS*

20News-M5 Doc2Vec						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.86**	0.76	0.60	0.57	0.64	0.52
NMI	**0.84**	0.77	0.66	0.65	0.70	0.63
FMI	**0.89**	0.80	0.68	0.65	0.71	0.61

(e) *Doc2Vec*

20News-M5 NASARI						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.79**	0.77	0.65	0.61	0.57	0.59
NMI	**0.81**	0.80	0.71	0.71	0.68	0.71
FMI	**0.83**	0.82	0.72	0.68	0.66	0.67

(f) *NASARI*

significant drop in performance on *20News-C10*. This happens because of the semantic overlaps. For example, all methods are weak on distinguishing the category

20News-M5 Sent2Vec						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	0.56	0.53	**0.72**	0.70	0.65	0.46
NMI	0.67	0.64	**0.74**	0.74	0.70	0.58
FMI	0.67	0.64	**0.77**	0.76	0.71	0.56

(g) *Sent2Vec*

20News-M5 LexVec						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	0.85	**0.86**	0.73	0.67	0.59	0.55
NMI	0.85	**0.88**	0.79	0.78	0.70	0.71
FMI	**0.88**	0.88	0.79	0.73	0.67	0.64

(h) *LexVec*

20News-M5 fastText						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	0.64	**0.77**	0.68	0.68	0.65	0.56
NMI	0.68	**0.77**	0.72	0.74	0.75	0.70
FMI	0.71	**0.81**	0.74	0.74	0.72	0.65

(i) *fastText*

20News-M5 GloVe						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	0.80	**0.82**	0.83	0.72	0.58	0.55
NMI	**0.83**	0.82	0.83	0.78	0.69	0.70
FMI	0.84	**0.86**	0.86	0.77	0.66	0.63

(j) *GloVe*

Table 5.0: Scores for each document clustering method on *20News-M5* setting the number of desired clusters $\boldsymbol{K} = 5, 6, 7, 8, 9$. **(a)** for *DSCTP*; **(b)** for *APV-BRSC*; **(c)** for *APV-TRSC*; **(d)** for *GPGS*; **(e)** for *Doc2Vec*; and **(f)** for *NASARI*; **(g)** for *Sent2Vec*; **(h)** for *LexVec*; **(i)** for *fastText* and **(j)** for *GloVe*. The highest scores in each sub-table are in bold.

20News-M5 K=5										
	DSCTP	**APV-BRSC**	**APV-TRSC**	**GPGS**	**Doc2vec**	**NASARI**	**fastText**	**Sent2Vec**	**LexVec**	**GloVe**
ARI	**0.90**	0.84	0.87	0.88	0.86	0.79	0.64	0.56	0.85	0.80
NMI	**0.91**	0.86	0.87	0.87	0.84	0.81	0.68	0.67	0.85	0.83
FMI	**0.92**	0.87	0.90	0.90	0.89	0.83	0.71	0.67	0.88	0.84

(a) Scores for $\boldsymbol{K} = 5$.

20News-M5 K=5,6,7,8,9,10										
	DSCTP	**APV-BRSC**	**APV-TRSC**	**GPGS**	**Doc2vec**	**NASARI**	**fastText**	**Sent2Vec**	**LexVec**	**GloVe**
ARI	**0.90**	0.84	0.87	0.88	0.86	0.79	0.77	0.72	0.86	0.82
NMI	**0.91**	0.86	0.87	0.87	0.84	0.81	0.77	0.74	0.88	0.83
FMI	**0.92**	0.87	0.90	0.90	0.89	0.83	0.81	0.77	0.88	0.86

(b) Highest scores for $\boldsymbol{K} = 5, 6, 7, 8, 9, 10$.

Table 5.1: Scores for *20News-M5*. **(a)** shows the scores for $\boldsymbol{K} = 5$; and **(b)** shows the best scores for $\boldsymbol{K} = 5, 6, 7, 8, 9, 10$.

"comp.sys.ibm.pc.hardware" from the category *"misc.forsale"*.

Tables 5.4 and 5.5 show the results for *Reuters-M7*. Once again, *DSCTP* outperforms all others, and by a more significant amount, on this dataset. *APV-BRSC*, *APV-TRSC* and *GPGS* perform as well as most of the other state-of-the-art methods and outperform *Doc2Vec*. *APV-BRSC* has its best *ARI* at $K = 10$ and $K = 11$; however, the *ARI* score at $K = 8$ is not far from the best score. Similarly, though the best *FMI* score of *APV-BRSC* is reached at $K = 10$ and $K = 11$, the score at $K = 8$ is nearly the same. The best *NMI* score of *APV-BRSC* occurs at $K = 8$ and $K = 9$. Thus, *APV-BRSC* can be considered to give its best performance at $K = 8$. *APV-BRSC* receives an explicit improvement from $K = 7$ to $K = 8$. We observe that at $K = 7$ *APV-BRSC* has difficulty distinguishing *"crude"*, *"livestock"*, *"ship"* and *"soybean"*, and hence there is a resulting cluster containing a "mix-up" of documents from these categories. At $K = 8$, *APV-BRSC* successfully detects *"crude"* thereby separating the documents in this category out of the "mix-up" cluster and creating an individual cluster for them. *APV-TRSC* reaches its best scores at $K = 12$, though the scores at $K = 9$ are close to the best. When looking into the resulting clustering of *APV-TRSC* at $K = 7$, we observe that there is also a "mix-up" cluster similar to that in *APV-BRSC*'s clusterings. *APV-TRSC* exhibits an improvement from $K = 7$ to $K = 9$ because it distinguishes the documents of *"ship"* from the "mix-up" cluster. *GPGS* reaches its best scores at $K = 9$. Similar to *APV-BRSC* and *APV-TRSC*,

20News-C10 DSCTP											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	**0.64**	0.63	0.56	0.48	0.46	0.44	0.43	0.40	0.45	0.45	0.45
NMI	**0.73**	0.72	0.69	0.64	0.62	0.62	0.64	0.60	0.64	0.65	0.64
FMI	**0.68**	0.67	0.60	0.53	0.51	0.49	0.49	0.45	0.50	0.50	0.50

(a) *DSCTP*

20News-C10 APV-BRSC											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	0.36	0.28	0.28	0.27	**0.38**	0.27	0.35	0.34	0.33	0.30	0.30
NMI	**0.59**	0.54	0.54	0.54	0.59	0.53	0.58	0.57	0.57	0.55	0.57
FMI	**0.46**	0.41	0.40	0.39	0.46	0.37	0.42	0.41	0.40	0.38	0.37

(b) *APV-BRSC*

20News-C10 APV-TRSC											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	**0.52**	0.41	0.45	0.46	0.46	0.42	0.43	0.41	0.36	0.42	0.40
NMI	**0.67**	0.62	0.62	0.63	0.64	0.61	0.62	0.61	0.59	0.62	0.60
FMI	**0.57**	0.49	0.52	0.52	0.52	0.49	0.49	0.47	0.43	0.47	0.46

(c) *APV-TRSC*

20News-C10 GPGS											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	0.46	**0.51**	0.50	0.48	0.47	0.48	0.49	0.46	0.46	0.45	0.42
NMI	**0.65**	0.65	0.64	0.63	0.62	0.64	0.64	0.63	0.63	0.62	0.62
FMI	0.53	**0.56**	0.55	0.54	0.52	0.53	0.53	0.51	0.51	0.51	0.48

(d) *GPGS*

20News-C10 Doc2Vec											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	**0.47**	0.44	0.41	0.43	0.41	0.39	0.38	0.37	0.31	0.34	0.26
NMI	**0.56**	0.55	0.51	0.54	0.52	0.51	0.51	0.54	0.48	0.49	0.42
FMI	**0.52**	0.50	0.46	0.48	0.46	0.44	0.43	0.43	0.37	0.40	0.32

(e) *Doc2Vec*

20News-C10 NASARI											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	**0.57**	0.52	0.53	0.50	0.49	0.47	0.45	0.42	0.45	0.39	0.36
NMI	**0.66**	0.62	0.63	0.63	0.64	0.61	0.61	0.58	0.60	0.61	0.60
FMI	**0.61**	0.57	0.58	0.55	0.54	0.52	0.51	0.48	0.50	0.46	0.42

(f) *NASARI*

GPGS has a mix-up cluster at $K = 7$ primarily consisting of *'livestock'"* and *"soybean"*,

20News-C10 Sent2Vec											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	0.34	0.32	0.34	0.40	0.40	0.39	0.42	**0.44**	0.36	0.39	0.38
NMI	0.48	0.47	0.49	0.53	0.56	0.54	0.56	**0.57**	0.53	0.56	0.55
FMI	0.42	0.40	0.41	0.46	0.46	0.45	0.48	**0.49**	0.42	0.44	0.44

(g) *Sent2Vec*

20News-C10 LexVec											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	0.63	**0.64**	0.64	0.58	0.61	0.64	0.56	0.54	0.54	0.57	0.49
NMI	0.72	0.73	0.72	0.69	0.71	**0.75**	0.70	0.71	0.70	0.71	0.68
FMI	0.67	**0.68**	0.67	0.62	0.65	0.68	0.61	0.59	0.59	0.62	0.54

(h) *LexVec*

20News-C10 fastText											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	0.48	0.48	0.52	**0.59**	0.59	0.51	0.54	0.53	0.48	0.43	0.44
NMI	0.60	0.61	0.64	**0.71**	0.69	0.65	0.69	0.66	0.65	0.62	0.65
FMI	0.53	0.53	0.57	**0.63**	0.63	0.55	0.59	0.58	0.53	0.49	0.50

(i) *fastText*

20News-C10 GloVe											
K	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
ARI	0.53	**0.57**	0.53	0.57	0.54	0.46	0.49	0.46	0.43	0.42	0.40
NMI	0.65	0.67	0.65	**0.69**	0.66	0.65	0.63	0.63	0.62	0.61	0.60
FMI	0.58	**0.61**	0.57	0.61	0.59	0.51	0.54	0.52	0.49	0.47	0.45

(j) *GloVe*

Table 5.2: Scores for each document clustering method on *20News-C10* setting the number of desired clusters $\boldsymbol{K} = 10, 11, 12, 13, 14$. **(a)** for *DSCTP*; **(b)** for *APV*; **(c)** for *GPGS*; **(d)** for *Doc2Vec*; and **(e)** for *NASARI*; **(g)** for *Sent2Vec*; **(h)** for *LexVec*; **(i)** for *fastText* and **(j)** for *GloVe*. The highest scores in each sub-table are in bold.

20News-C10 K=10										
	DSCTP	**APV-BRSC**	**APV-TRSC**	**GPGS**	**Doc2Vec**	**NASARI**	**fastText**	**Sent2Vec**	**LexVec**	**GloVe**
ARI	**0.64**	0.36	0.52	0.46	0.47	0.57	0.48	0.34	0.63	0.53
NMI	**0.73**	0.59	0.67	0.65	0.56	0.66	0.60	0.48	0.72	0.65
FMI	**0.68**	0.46	0.57	0.53	0.52	0.61	0.53	0.42	0.67	0.58

(a) Scores for $\boldsymbol{K} = 10$.

20News-C10 K=10,11,12,13,14,15,16,17,18,19,20										
	DSCTP	**APV-BRSC**	**APV-TRSC**	**GPGS**	**Doc2Vec**	**NASARI**	**fastText**	**Sent2Vec**	**LexVec**	**GloVe**
ARI	**0.64**	0.38	0.52	0.51	0.47	0.57	0.59	0.44	**0.64**	0.57
NMI	0.73	0.59	0.67	0.65	0.56	0.66	0.71	0.57	**0.75**	0.69
FMI	**0.68**	0.46	0.57	0.56	0.52	0.61	0.63	0.49	**0.68**	0.61

(b) Best scores for $\boldsymbol{K} = 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20$.

Table 5.3: Scores for *20News-C10*. **(a)** shows the scores for $\boldsymbol{K} = 10$; and **(b)** shows the best scores for $\boldsymbol{K} = 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20$.

and at $K = 9$ *GPGS* separates them.

Table 5.6 and 5.7 show the results for **BBC-M5**. On this dataset, *DSCTP* also gives a top performance, though *Sent2Vec* performs a slightly better than *DSCTP*. *APV-BRSC*, *APV-TRSC* and *GPGS* outperform *Doc2Vec* and *fastText*, and perform similarly to the remaining methods. *APV-BRSC* encounters some difficulties in distinguishing several "*business*" and "*entertainment*" documents from "*tech*" documents. *APV-TRSC* has some difficulties in distinguishing several "*business*" and "*entertainment*" documents from "*politics*" documents. *GPGS* has major trouble on "*politics*" and "*sport*".

5.6 Discussion

We investigate a particular observation from our experimental results. That is, *DSCTP* outperforms most of the methods on each of the three datasets, whereas *APV-BRSC*, *APV-TRSC* and *GPGS*, though based on *DSCTP*, do not perform as well as *DSCTP*.

Recall that *APV* uses spectral clustering (with normalized Laplacian), and spectral clustering actually finds an optimal solution to minimize an objective function [157]:

$$\mathrm{NCut}(C_1, \ldots, C_K) = \sum_{i=1}^{K} \frac{Cut(C_i, \overline{C_i})}{Vol(C_i)} \tag{5.6}$$

Reuters-M7 DSCTP								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	**0.90**	0.87	0.82	0.75	0.71	0.67	0.61	0.62
NMI	**0.91**	0.88	0.85	0.80	0.79	0.77	0.74	0.76
FMI	**0.92**	0.89	0.84	0.79	0.76	0.72	0.67	0.68

(a) *DSCTP*

Reuters-M7 APV-BRSC								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	0.46	0.57	0.58	**0.59**	0.59	0.56	0.54	0.56
NMI	0.64	**0.71**	0.71	0.70	0.69	0.68	0.66	0.66
FMI	0.55	0.64	0.64	**0.65**	0.65	0.62	0.60	0.62

(b) *APV-BRSC*

Reuters-M7 APV-TRSC								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	0.46	0.46	0.50	0.44	0.43	**0.53**	0.37	0.37
NMI	0.64	0.64	0.66	0.63	0.63	**0.67**	0.59	0.59
FMI	0.55	0.55	0.58	0.53	0.53	**0.60**	0.47	0.47

(c) *APV-TRSC*

Reuters-M7 GPGS								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	0.57	0.53	**0.69**	0.57	0.58	0.56	0.56	0.54
NMI	0.70	0.68	**0.76**	0.72	0.71	0.69	0.69	0.68
FMI	0.64	0.61	**0.73**	0.63	0.64	0.62	0.62	0.61

(d) *GPGS*

Reuters-M7 Doc2Vec								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	0.40	0.41	0.40	**0.42**	0.41	0.42	0.35	0.45
NMI	0.52	0.53	0.51	0.54	0.51	**0.55**	0.52	0.59
FMI	0.49	**0.50**	0.48	0.50	0.49	0.50	0.43	0.52

(e) *Doc2Vec*

Reuters-M7 NASARI								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	0.61	0.61	**0.62**	0.58	0.59	0.56	0.56	0.50
NMI	0.67	0.67	0.69	0.68	**0.70**	0.70	0.68	0.68
FMI	0.66	0.66	**0.67**	0.64	0.65	0.62	0.62	0.58

(f) *NASARI*

where $\{C_i\}$ denotes a partitioning of a given dataset X, $Cut(C_i, \overline{C_i})$ denotes the cost

Reuters-M7 Sent2Vec								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	**0.75**	0.75	0.72	0.63	0.67	0.60	0.58	0.57
NMI	**0.79**	0.78	0.77	0.71	0.75	0.72	0.71	0.70
FMI	**0.78**	0.78	0.76	0.68	0.71	0.66	0.64	0.63

(g) *Sent2Vec*

Reuters-M7 LexVec								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	0.58	**0.59**	0.55	0.50	0.45	0.53	0.42	0.47
NMI	**0.64**	0.64	0.64	0.62	0.61	0.67	0.58	0.65
FMI	**0.64**	0.64	0.61	0.56	0.52	0.59	0.50	0.54

(h) *LexVec*

Reuters-M7 fastText								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	0.52	0.49	**0.64**	0.62	0.64	0.64	0.59	0.52
NMI	0.62	0.59	**0.70**	0.70	0.73	0.71	0.68	0.66
FMI	0.59	0.56	**0.69**	0.67	0.69	0.69	0.65	0.59

(i) *fastText*

Reuters-M7 GloVe								
K	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ARI	0.64	**0.71**	0.67	0.67	0.63	0.63	0.63	0.60
NMI	0.69	**0.75**	0.72	0.74	0.74	0.73	0.74	0.74
FMI	0.69	**0.75**	0.71	0.72	0.69	0.69	0.68	0.66

(j) *GloVe*

Table 5.4: Scores for each document clustering method on *Reuters-M5* setting the number of desired clusters $\boldsymbol{K} = 7, 8, 9, 10, 11$. **(a)** for *DSCTP*; **(b)** for *APV-BRSC*; **(c)** for *APV-TRSC*; **(d)** for *GPGS*; **(e)** for *Doc2Vec*; **(f)** for *NASARI*; **(g)** for *Sent2Vec*; **(h)** for *LexVec*; **(i)** for *fastText* and **(j)** for *GloVe*. The highest scores in each sub-table are in bold.

Reuters-M7 K=7										
	DSCTP	**APV-BRSC**	**APV-TRSC**	**GPGS**	**Doc2vec**	**NASARI**	**fastText**	**Sent2Vec**	**LexVec**	**GloVe**
ARI	**0.90**	0.46	0.46	0.57	0.40	0.61	0.52	0.75	0.58	0.64
NMI	**0.91**	0.64	0.64	0.70	0.52	0.67	0.62	0.79	0.64	0.69
FMI	**0.92**	0.55	0.55	0.64	0.49	0.66	0.59	0.78	0.64	0.69

(a) Scores for $\boldsymbol{K} = 7$.

Reuters-M7 K=7,8,9,10,11,12,13,14										
	DSCTP	**APV-BRSC**	**APV-TRSC**	**GPGS**	**Doc2vec**	**NASARI**	**fastText**	**Sent2Vec**	**LexVec**	**GloVe**
ARI	**0.90**	0.59	0.53	0.69	0.42	0.62	0.64	0.75	0.55	0.71
NMI	**0.91**	0.71	0.67	0.76	0.55	0.70	0.73	0.79	0.67	0.75
FMI	**0.92**	0.65	0.60	0.73	0.50	0.67	0.69	0.78	0.61	0.75

(b) Highest scores for $\boldsymbol{K} = 7, 8, 9, 10, 11, 12, 13, 14$.

Table 5.5: Scores for *Reuters-M7*. **(a)** shows the scores for $\boldsymbol{K} = 7$; and **(b)** shows the highest scores for $\boldsymbol{K} = 7, 8, 9, 10, 11, 12, 13, 14$.

of generating a partition C_i and $Vol(C_i)$ denotes the *volume* of C_i. In our document clustering methods, if a clustering is of a high quality (as evaluated by *ARI*, *NMI* and *FMI*), then naturally it implies that the best solution to *NCut* found by spectral clustering largely coincides with the ground truth classes. Then a reason that *APV-BRSC* and *APV-TRSC* do not perform as well as *DSCTP* must be that spectral clustering could not find solutions with these methods that are as good as that found with *DSCTP*. Thus, there is at least one pair of categories that these methods are weak at, though different methods may be weak at different pairs. If a method is weak at distinguishing a pair of categories, then in its clustering there is a subset of documents belonging to a category which are clustered into the other category. Then these mis-clustered documents must have relatively low similarities with other documents in that other category. How could this happen? To find concrete facts to demonstrate our explanations, we conducted another two experiments for *APV-BRSC*[19]. First, we applied *APV-BRSC* to a dataset consisting of the documents from the *"soybean"* and *"livestock"* categories of *Reuters-M7*. We chose these two categories because *APV-BRSC* is confused on them (according to our experimental results on *Reuters-M7*). We kept all settings the same as for *Reuters-M7* except setting the desired number of clusters to 2. In the resulting clustering, 20 *"livestock"* documents

[19] *APV-TRSC* has the same issue as *APV-BRSC*.

BBC-M5 DSCTP						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.88**	0.85	0.74	0.70	0.67	0.51
NMI	**0.86**	0.86	0.75	0.76	0.77	0.66
FMI	**0.90**	0.88	0.79	0.76	0.74	0.61

(a) *DSCTP*

BBC-M5 APV-BRSC						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.76**	0.73	0.65	0.61	0.56	0.59
NMI	**0.76**	0.74	0.69	0.67	0.66	0.66
FMI	**0.80**	0.78	0.72	0.68	0.64	0.66

(b) *APV-BRSC*

BBC-M5 APV-TRSC						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.70**	0.67	0.68	0.51	0.45	0.31
NMI	**0.71**	0.68	0.69	0.70	0.61	0.55
FMI	**0.76**	0.74	0.74	0.60	0.55	0.43

(c) *APV-TRSC*

BBC-M5 GPGS						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.64**	0.62	0.54	0.51	0.46	0.43
NMI	**0.71**	0.68	0.66	0.66	0.63	0.60
FMI	**0.72**	0.69	0.63	0.60	0.56	0.53

(d) *GPGS*

BBC-M5 Doc2Vec						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.55**	0.48	0.47	0.32	0.29	0.40
NMI	**0.57**	0.52	0.49	0.43	0.39	0.48
FMI	**0.64**	0.58	0.57	0.44	0.41	0.50

(e) *Doc2Vec*

BBC-M5 NASARI						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.72**	0.64	0.58	0.57	0.53	0.42
NMI	**0.73**	0.68	0.63	0.66	0.65	0.56
FMI	**0.78**	0.70	0.66	0.65	0.62	0.52

(f) *NASARI*

are in one cluster (we call this cluster C_1), and the remaining documents are in the

BBC-M5 Sent2Vec						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.89**	0.77	0.73	0.64	0.48	0.55
NMI	**0.88**	0.79	0.78	0.72	0.67	0.73
FMI	**0.91**	0.82	0.78	0.71	0.58	0.64

(g) *Sent2Vec*

BBC-M5 LexVec						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.82**	0.72	0.62	0.57	0.53	0.51
NMI	**0.81**	0.74	0.69	0.67	0.67	0.65
FMI	**0.85**	0.77	0.69	0.65	0.62	0.60

(h) *LexVec*

BBC-M5 fastText						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	0.60	**0.64**	0.60	0.51	0.51	0.47
NMI	0.64	**0.67**	0.63	0.63	0.62	0.63
FMI	0.68	**0.71**	0.67	0.60	0.60	0.57

(i) *fastText*

BBC-M5 GloVe						
K	**5**	**6**	**7**	**8**	**9**	**10**
ARI	**0.80**	0.71	0.57	0.55	0.48	0.43
NMI	**0.80**	0.74	0.68	0.66	0.64	0.64
FMI	**0.84**	0.76	0.65	0.63	0.58	0.54

(j) *GloVe*

Table 5.6: Scores for each document clustering method on *Reuters-M5* setting the number of desired clusters $\boldsymbol{K} = 7, 8, 9, 10, 11$. **(a)** for *DSCTP*; **(b)** for *APV-BRSC*; **(c)** for *APV-TRSC*; **(d)** for *GPGS*; **(e)** for *Doc2Vec*; and **(f)** for *NASARI*; **(g)** for *Sent2Vec*; **(h)** for *LexVec*; **(i)** for *fastText* and **(j)** for *GloVe*. The highest scores in each sub-table are in bold.

BBC-M5 K=5										
	DSCTP	**APV-BRSC**	**APV-TRSC**	**GPGS**	**Doc2Vec**	**NASARI**	**fastText**	**Sent2Vec**	**LexVec**	**GloVe**
ARI	0.88	0.76	0.70	0.64	0.55	0.72	0.60	**0.89**	0.82	0.80
NMI	0.86	0.76	0.71	0.71	0.57	0.73	0.64	**0.88**	0.81	0.80
FMI	0.90	0.80	0.76	0.72	0.64	0.78	0.68	**0.91**	0.85	0.84

(a) Scores for $\boldsymbol{K} = 5$.

BBC-M5 K=5,6,7,8,9,10										
	DSCTP	**APV-BRSC**	**APV-TRSC**	**GPGS**	**Doc2Vec**	**NASARI**	**fastText**	**Sent2Vec**	**LexVec**	**GloVe**
ARI	0.88	0.70	0.76	0.64	0.55	0.72	0.64	**0.89**	0.82	0.80
NMI	0.86	0.71	0.76	0.71	0.57	0.73	0.67	**0.88**	0.81	0.80
FMI	0.90	0.76	0.80	0.72	0.64	0.78	0.71	**0.91**	0.85	0.84

(b) Highest scores for $\boldsymbol{K} = 5, 6, 7, 8, 9, 10$.

Table 5.7: Scores for *BBC-M5*. **(a)** shows the scores for $\boldsymbol{K} = 5$; and **(b)** shows the highest scores for $\boldsymbol{K} = 5, 6, 7, 8, 9, 10$.

other cluster which thereby consists of 30 *"livestock"* and 50 *"soybean"* documents (we call this cluster C_2). Focusing on the *"livestock"* documents, we observe that the average semantic similarity of pairs of *"livestock"* documents when using *APV-BRSC* representation and cosine similarity is 0.10 and the standard deviation is 0.19. Then, looking at the similarities of pairs of *"livestock"* documents across C_1 and C_2 (i.e. one document in C_1 and the other in C_2), there are in total 172 of these similarities out of 600 that are lower than 0.0050 (which is obviously a super low value relative to the average 0.10). Among these 172 pairs of documents, consider a typical example: *"livestock/12376"* and *"livestock/3278"*. The similarity between them given by *APV-BRSC* and cosine similarity is 0.0036 while the similarity given by *DSCTP* is 215.8238. The average similarity of pairs of *"livestock"* documents by *DSCTP* is 237.1849. Then 215.8238 is a significant similarity value. To better understand the reasons of the large difference of similarities, we consider the *GPs* extracted from the minimum basic cycles and the *APs* into which these GPs are clustered. We found that with *APV-BRSC* many *GPs* in the same minimum basic cycles are clustered into different *APs*. This is an issue that we pointed out at the end of Chapter 3. Here we provide a concrete example. In *"livestock/12376"*, a sentence contains two words: *poultry* and *facilities*; and in *"livestock/3278"*, a sentence contains: *cattle* and *contracts*. Specifically, *poultry* is determined to be similar to *cattle*, and *facilities* is determined to be similar to *contracts*. These four words form a minimum basic cycle, and {*poultry*, *facilities*} and {*cattle*, *contracts*} are two *GPs*[20]. In *DSCTP*, this minimum basic cycle, as a whole, contributes a portion of the similarity between the two documents. However, in *APV-BRSC*, since the two *GPs* are clustered into different *APs*, they will contribute to different dimensions of the *APV*. This phenomenon undermines the contribution of the similarity between the two *GPs* to the document semantic similarity.

In a similar way to the above, we apply *GPGS* to a dataset that only contains documents in *"ship"* and *"crude"*. The result is that 39 *"ship"* documents are in one cluster (called D_1) and the remainder (10 *"ship"* and 50 *"crude"*) are in the other (called D_2). Similar to the *APV-BRSC* case, this clustering happens because the document similarities across D_1 and D_2 are relatively weak. When looking into the *GPGS* representations, it turns out that in a *GPGS*, induced significance values may be overestimated. Here is a sample pair of documents to demonstrate the issue. Consider the documents *"ship/4778"* and *"ship/3329"*. The similarity between them as computed using our *GPGS* representation and cosine similarity is 0.0044, while the similarity given by *DSCTP* is 180.2122. How could this happen? In the *GPGSs* of *"ship/4778"* and *"ship/3329"* respectively, there is a shared *GP*: {*shipping*, *cargo*}. This *GP* is assigned 9.36 as its significance value in the *GPGSs* of *"ship/4778"*, and is assigned 517.67 in the *GPGSs* of *"ship/3329"*. The average significance value in the *GPGSs* of *"ship/4778"* is 88.69, and the average significance value in the *GPGSs* of *"ship/3329"* is 69.41. However, neither of these two documents contains this *GP*. That is, the significance values of this *GP* in the two *GPGSs* are induced values. This example shows that induced values can sometimes be large, and large induced values may lead to significant differences at some dimensions which will further lead to significant deviation in document similarity.

These issues and observations provide motivation for future work to find document representations that work as well as *DSCTP* on *HDC* tasks.

[20] We temporarily ignore the shortest path lengths of these two *GPs*.

Chapter 6

MESSAGE PROPAGATION PROBLEM

6.1 Introduction

Social Dynamics [41] is a rapidly developing area especially as pertains to social networks (e.g. *Twitter*). In 2017, *DARPA*[1] in conjunction with the Biocomplexity Institute of Virginia Tech[2] started a project (named *SocialSim* [113]) aimed at simulating and detecting social behaviors in social networks. In 2018 summer, the author of this dissertation participated in this project. In this chapter, we outline utilizing *DSCTP* and the *DSRs* discussed in Chapters 3 and 4 to provide and improve the simulation techniques for this project. Particularly, we focus on a featured problem of this project and explain how our methods will be used to address this problem.

The featured problem is named the **message propagation (MP) problem** and studies how messages are propagated in social networks. In a general way, the problem can be stated as follows:

Message Propagation (MP) Problem:

Given a *message*, a specified user who initially issues this *message*, and a social network, the problem asks how likely it is that each user in the social network will *propagate* this *message*.

To formulate this problem concretely, two key concepts need to be clarified, namely *message* and *propagate*.

[1] Defense Advanced Research Projects Agency: https://www.darpa.mil

[2] In 2019, the relevant departments of the Biocomplexity Institute of Virginia Tech moved to the University of Virginia, and a new institute also named Biocomplexitiy was founded. The *SocialSim* project was then continued at the University of Virginia.

Def: Message:

A **message** in this problem is a piece of human understandable text in English, and is always issued by one and only one user in the social network.

A typical example of message is *tweets* on *Twitter*. Such a message can contain one or multiple sentences or a number of words. Regardless, every message has a common-sense *semantics* (as described in Chapter 1), and the semantics of a message can be represented as an APV or as a $GPGS$ respectively.

Def: Propagate:

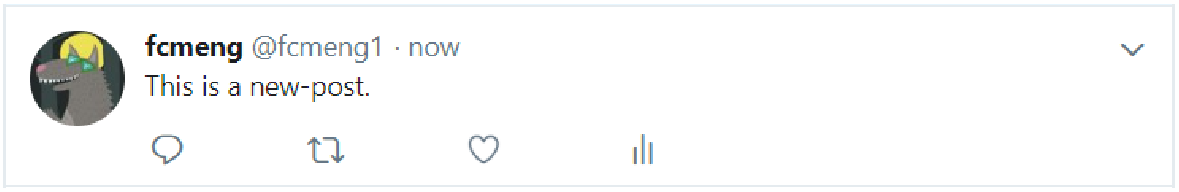
A message m_e is said to be **propagated** by a user u_r, if u_r issues another message m_r as a reaction to m_e after seeing m_e. Such a reaction is called a **message propagation activity (MPA)**.

In social networks, there exist a number of *MPAs*. For example, *new-post*, *re-post*, *quote*, *at (i.e @)*, and *push messages* are all commonly seen *MPAs*. Some of these are user behaviors (e.g. *re-post*), while some are machine behaviors (e.g. *push messages*)[3]. Amongst these *MPAs*, a subset of them are more fundamental than others (i.e. those activities typically supported by most social networks). In this work we focus on three *MPAs* namely *new-post*, *re-post* and *quote* are such fundamental ones.

- When a user issues a new message (e.g. a *tweet*), this activity is said to be a **new-post**. An example is shown in Figure 6.1. A *new-post* activity can be a consequence of a user seeing a message issued by another user.[4] For example, if one sees that a friend tweeted a fun limerick, then he may want to post a limerick as well, yet without *quoting* his friend's *tweet*. In this case, a *new-post* is a *MPA*.

[3] Robots in social networks are considered as users.

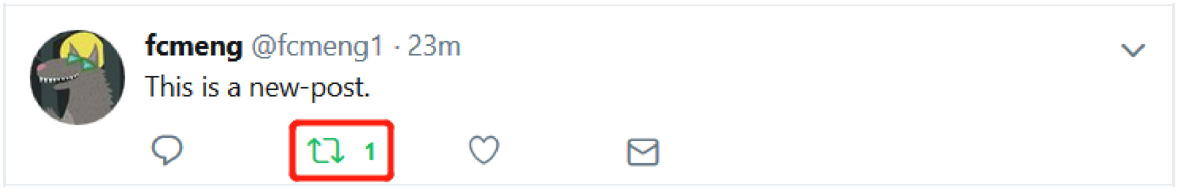
[4] Of course, a *new-post* activity can be originated by a user with seeing a prior message, but that it is not the case that interests this work.


fcmeng @fcmeng1 · now

This is a new-post.

Figure 6.1: A *new-post* by the user *fcmeng* on *Twitter* with the text "*This is a new-post.*"

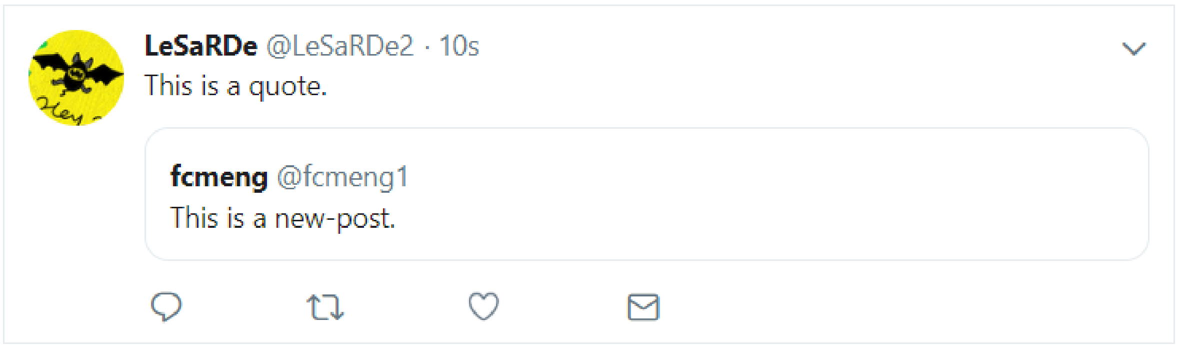
- When a user issues a new message containing an existing message without any additional comment, it is said to be a **re-post**. An example is shown in Figure 6.2.


fcmeng @fcmeng1 · 23m

This is a new-post.

Figure 6.2: A *tweet* issued by *fcmeng* is *re-posted* once (highlighted by the *red rectangle*).

- When a user issues a new message containing an existing message with an additional comment, it is said to be a **quote**. An example is shown in Figure 6.3


LeSaRDe @LeSaRDe2 · 10s

This is a quote.

fcmeng @fcmeng1

This is a new-post.

Figure 6.3: A *tweet* issued by *fcmeng* is *quoted* by another user *LeSaRDe* with a comment "*This is a quote.*"

It is important to note that a message m_r from *new-post* or *quote* can have a significantly different *semantics* from the message m_e seen by the user who issues m_r. Thus, a key question to be answered is:

What is actually propagated when a message is said to be propagated?

The answer, in our work is the *semantics of the message*. That means that m_r and m_e have to be semantically similar so that it is affirmative that the content of m_e (i.e. what the issuer of m_e wants to convey) has been conveyed to the user who issues m_r. Such *propagations* are said to be **effective**, and defined as follows:

Def: Effective Propagation:

Let m_e denote a message seen by a user, and let m_r denote the next message issued by that user after seeing m_e. Also, let $d_s : \mathbb{M} \times \mathbb{M} \mapsto \mathbb{R}_{\geq 0}$ denote a semantic distance function, where $\mathbb{M}$ denotes the set of messages. The issuing of m_r is said to be an **effective propagation** of m_e, if $d_s(m_e, m_r) \leq \theta_m$, where θ_m is a predetermined threshold for semantic distances.

Thus, returning to the definition of the *MP* problem, in our work the *propagation* of a message means the *effective propagation* of the message.

Our study of the *MP* problem is based on a simple social network model which is introduced in Section 6.2. The *MP* problem under this simplified scenario is formulated in Section 6.3, also therein a message propagation analysis method is proposed.

6.2 A Simple Social Network Model

In this section, a **simple social network model (SSNM)** is introduced. This scenario specifies a set of fundamental elements to form a minimal social network. The formal definition is given in Section 6.2.1. In Section 6.2.2, the *message propagation* in *SSNM* is discussed.

6.2.1 Concepts & Definitions

To define the *SSNM*, the relationships between users in a social network need to be formalized. Motivated by the two most fundamental user relationships in social networks, *follow* and *friend*, a generalized user relationship named the **following relationship** is defined as follows:

Def: Following Relationship:

Given two distinct users in a social network, u_e and u_r, they are said to have a **following relationship (FR)**, denoted by $u_e \rightarrow u_r$, if u_r *follows* or *friends* u_e. In this *FR*, u_e is called the **followee**, and u_r is called the **follower**. Also, if they have such a *FR*, then all of the messages issued by u_e are directly *exposed* to u_r. Users are not allowed to establish a *FR* with themselves.

It is important to note that *expose* is not the same as *see*, because even though a message issued by B is available to A, A may not actually read this message. In this case, we say that this message is **exposed** to A but not **seen** by A.

The *SSNM* is defined as:

Def: Simple Social Network Model (SSNM):

An **SSNM** consists of a finite set of users (who can issue messages), a set of *FRs* between the users, and a set of *MPAs* including *new-post*, *re-post* and *quote*, and two *non-MPA* reactions, *miss* and *ignore* defined as:

When a message m is exposed to a user u, it is possible that u fails to see that message. Then u failing to see m is considered a reaction of u to m. This reaction is called a **miss**. When u does see m, but chooses to react by doing nothing, this behavior is also considered a reaction of u to m and is called an **ignore**.

In an *SSNM*, a message m can only be *propagated* along *FRs* (i.e. if m is *propagated* by a user u_r, then m must be issued by another user u_e such that a *FR* $u_e \rightarrow u_r$ exists).

6.2.2 Message Propagation in SSNM

In an *SSNM*, the propagation of a message is formed by a series of *MPAs*. To formalize the message propagation, terminal conditions are needed. As discussed earlier, since the users in the *SSNM* are related to each other by *FRs*, and messages can only propagated along *FRs*, then intuitively, a message propagation can be viewed as a set of directed paths consisting of a subset of *FRs*, along which the message is propagated by *MPAs*. Then naturally for such a path, if a *miss* or an *ignore* occurs, this path would not propagate the message beyond the point of the *miss* or *ignore*. There is a more subtle propagation termination, namely that *MPAs* may not preserve the exact semantics of the original message in their propagations (even though the propagations are effective). Then, it is likely that beyond a certain propagation the semantics would deviate from the acceptable range. Then for a path, if an ineffective propagation occurs, this path is terminated. It may happen that the semantics returns back to the acceptable range in the course of message propagation after some ineffective propagations emerge in between. This subtle situation is not considered in this work. Marking the message propagation as terminated whenever an ineffective propagation occurs is most reasonable. Thus, we define the **termination** of a message propagation as follows:

Def: Message Propagation Termination:

In an *SSNM*, only *MPAs* which result in effective propagations will propagate messages (i.e. *miss*, *ignore*, and *MPAs* which result in ineffective propagations do not propagate messages). For a given message, if no user performs *MPAs* resulting in effective propagations, then the propagation of the message is said to be **terminated**.

Users who react with *miss*, *ignore*, and *MPAs* which result in ineffective propagations are said to be at the **boundary** of the message propagation.

6.2.3 Difficulties in SSNM

Although an *SSNM* is already a simplified social network scenario, some of its characteristics may obstruct computational tasks from being applied to it. In this section, we first discuss in detail an *SSNM* example, and then list three of its characteristics that may cause difficulties in addressing the *MP* problem under the *SSNM*.

An example *SSNM* is shown in Figure 6.4. A is the initial user who issues a message m_0. B is one of the *followers* of A. B propagates m_0 with a probability. This probability is denoted by $P(r, q, n, A, B, m_0)$, where r denotes *re-post*, q denotes *quote* and n denotes *new-post*. Also, B *ignores* or *misses* m_0 with the probability $1 - P(r, q, n, A, B, m_0)$. The resulting message, if any, from B's propagation reaction is denoted by m_1. If the distance of the semantics between m_0 and m_1, denoted by $d_s(m_0, m_1)$, is less than a predetermined threshold θ_m, then the message m_1 is propagated to B's *followers* (e.g. C). Otherwise, the propagation of m_0 will not move forward beyond B though B still sends out the message. In this case, B is at the *boundary* of the propagation of m_0, and contributes nothing to the message propagation.

In addition, if a user propagates the message more than once (i.e. this user acts as a *follower* multiple times in the course of message propagation), then this user is said to be a **recurrent follower**. In Figure 6.5, B is a *recurrent follower*. The dashed line indicates that the message m_2 can be propagated from C to B. The probability of B propagating the message (no matter whether it is based on m_0 or m_2) is $P(r, q, n, A, m_0, C, m_2, B)$. This probability can be generalized to

$$P(r, q, n, u_{e1}, m_{e1}, \ldots, u_{en}, m_{en}, B)$$

where each u_{ei} denotes a *followee* of B, and m_{ei} denotes the message issued by u_{ei} in the propagation of m_0. The existence of *recurrent followers* may widen the range of the message propagation. Intuitively, when the number of recurrent followers increases, the likelihoods of the followers of those recurrent followers seeing the message will increase,

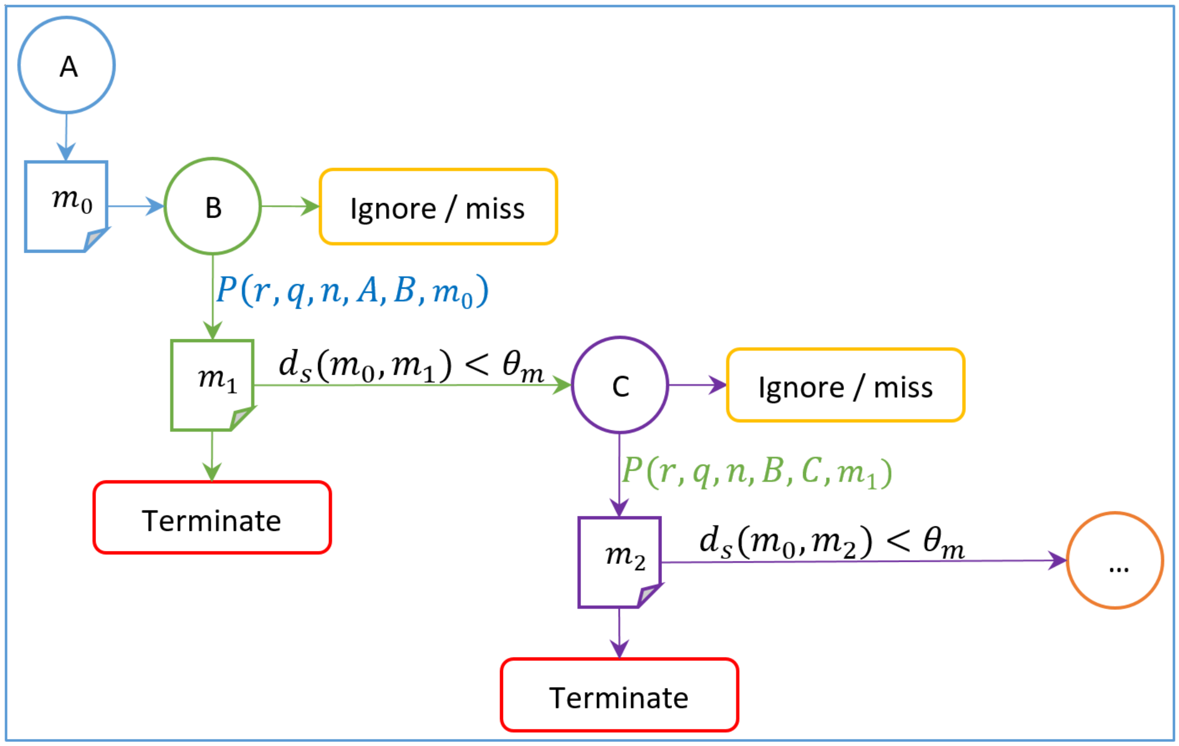


Figure 6.4: An example procedure of message propagation.

hence the likelihood of the message being propagated will increase, and consequently the propagation range will widen. In reality, such a range-widening can be witnessed in the case that a topic or an event is initially wildly discussed in a relatively small group of people, and then the discussion spreads virally.

Next, we discuss two issues with *SSNM*.

First, the probability of B reacting to m_0 issued by A with *re-post*, *quote* or *new-post* is $P(r,q,n,A,B,m_0) = P(r,q,n|A,B,m_0) \cdot P(A,B,m_0)$, where $P(A,B,m_0)$ is the probability of B seeing A issue m_0. However, estimating $P(A,B,m_0)$ is difficult without prior information. For example, it may depend on how frequently a user checks new messages and how many new messages emerge per day.

Second, as discussed in Section 6.2.1, in an *SSNM*, there are only *re-post*, *quote*, *new-post* and *ignore* as possible reactions of a *follower* when seeing a message. Then, $P(r,q,n|A,B,m_0) = 1 - P(i|A,B,m_0)$, where i denotes *ignore*. However, $P(i|A,B,m_0)$ can be difficult to compute, nor would computing $P(r,q,n|A,B,m_0)$ be easy, because

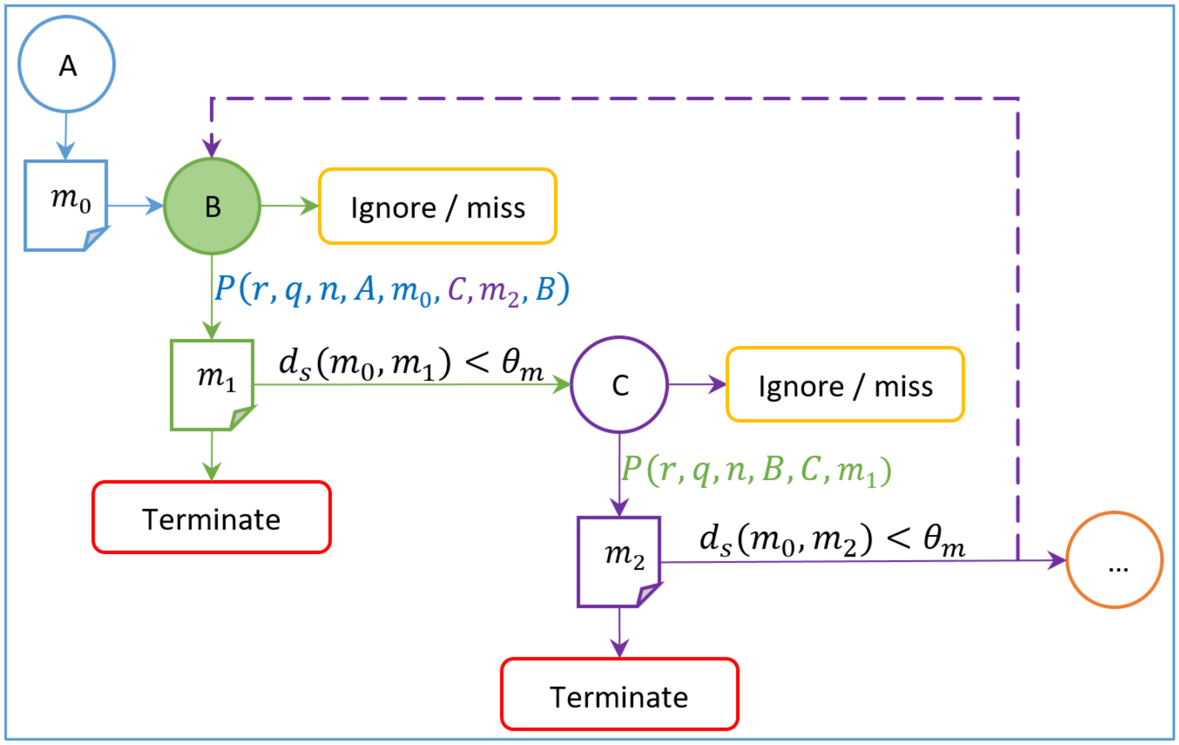


Figure 6.5: An example of *recurrent follower*. In this example, B is a recurrent follower.

of factors that are difficult to quantify. For example, if m_0 is one of B's favorite topics, then the probability of B propagating this message could be large; or, if B particularly prefers to *re-post* A's messages, then this probability could also be large. Taking all such factors into account to estimate $P(r, q, n|A, B, m_0)$ could be an arduous task.

Note that, in addition to the above, in an *SSNM*, the structure of the social network may not be fixed (i.e. the set of users and the set of *FRs* may vary). Theoretically, the message propagation and the variation of *FRs* can affect each other. Active users may attract more *followers* while inactive users may lose *followers*. On the other hand, the ones who attract more *followers* may contribute more to message propagations than others (i.e. more influential). These two dynamics form a recursive dynamic system. However, formulating this dynamic system would be difficult, and it deserves an individual study beyond the *MP* problem. Thus, in this work, we assume that the structure of a social network is fixed in an *SSNM*.

6.3 MP Problem Under SSNM

The *MP* problem under the *SSNM* is formulated in this section. As discussed above, some characteristics of the *SSNM* may lead to difficulties in addressing the *MP* problem. To overcome these obstacles, we introduce a series of assumptions relative to the *SSNM*, and based on these assumptions we formulate the *MP* problem under the *SSNM*. We name it the **simplified message propagation (SMP)** problem. The formulation including the assumptions is provided in Section 6.3.1. To address this *SMP* problem, we propose a potential solution based on document semantics.

6.3.1 Simplified Message Propagation Problem

Simplified Message Propagation (SMP) Problem:

Given:

- An *SSNM*.
- A message, m.
- A user u_0 in the *SSNM* who is the first one to issue m.

Assumptions:

a. Every *follower* will see every message issued by its *followees* with probability 1.

b. How a *follower* reacts (i.e. *quote*, *re-post*, *new-post* and *ignore*) when seeing a message depends only on the *follower* and the message.

c. A *follower*'s reaction message is always semantically similar to the given message.

Seek:

Each user in the *SSNM* is assigned a probability reflecting the likelihood of the user propagating the message m.

About the three assumptions:

The **assumption a** implies $P(u_e, u_r, m_e) = 1$, where u_e denotes a *followee*, u_r denotes a *follower* of u_e, and m_e is a message issued by u_e. From the discussion in Section 6.2.3, the probability of u_r propagating m_e is $P(r, q, n, u_e, u_r, m_e)$. Hence,

$$P(r, q, n, u_e, u_r, m_e) = P(r, q, n|u_e, u_r, m_e) \cdot P(u_e, u_r, m_e) \tag{6.1}$$

$$= P(r, q, n|u_e, u_r, m_e) \tag{6.2}$$

The **assumption b** implies that the probability $P(r, q, n|u_e, u_r, m_e)$ is large only when u_r is interested in the message m_e. For example, if m_e is a tweet about a pizza toppings, and food is a favorite topic of u_r, then it is very likely that u_r posts a tweet reacting to m_e.

The **assumption c** implies that the semantic distance between m_e and the reaction message m_r issued by u_r is always less than a predetermined threshold θ_m, (i.e. *MPAs* are always effective).

6.3.2 A Semantics-Based Idea

To attack the *SMP* problem, we propose a solution based on document semantics. This approach is based on two observations. One is that for two users in a *FR* (e.g. $u_e \to u_r$) the more semantically similar their message history, the more likely that u_r propagates the messages issued by u_e. The other is that when a user u_r sees a message m_e, the more m_e being semantically similar to u_r's messages, the more likely it is that u_r propagates m_e. Recall also that a message can only be propagated along *FRs*. Then combining the observations and the message propagation rule, a weighted and directed graph consisting of a subset of users and their *FRs* can naturally be utilized to reflect the message propagation. More specifically, directed edges reflect the message propagation directions, and edge weights reflect the likelihoods of *followers* propagating the message. We call such a graph a **message propagation graph (MPG)** for a given message. To formulate the *MPG*, we first introduce another graph, named the

FR graph (FRG) which consists of all *FRs* and the users in the *SSNM*. A *MPG* is a subgraph of a *FRG*.

Def: FR Graph (FRG):

Given a *SSNM*, the corresponding **FR graph** is a directed and weighted graph. Every user in the *SSNM* is associated to a vertex in the *FRG*. Every *FR* of two users in the *SSNM* is associated to a directed edge. The weight on an edge reflects the semantic similarity between two users' messages.

To construct a *FR* graph with respect to a *SSNM*, we need to compute the weight on each edge. To compute the weights, each user's messages (within a specific period) are collected (we call this collection the **message history** of this user) and treated as a whole (i.e. as one document), and *DSCTP* is utilized to compute the semantic similarity between the message histories of two users at the two endpoints of an edge. Then the weight of that edge is sat to be the semantic similarity value.

The *MPG* is defined below and an example is shown in Figure 6.6.

Def: Message Propagation Graph (MPG):

Given a *FRG* and a source vertex (i.e. a user who initially issues a given message m), a **message propagation graph** is a subgraph of the *FRG*. The subgraph contains all vertices of the *FRG* such that there exists at least one directed path from the source to each of these vertices. The *MPG* with respect to the source is induced by these vertices and all paths between the source and each of these vertices. Each edge weight (e.g. the weight on an edge $u_e \rightarrow u_r$) in the *MPG* reflects the likelihood (i.e. a probability) of the follower (i.e. u_r) propagating the message m when u_r sees m from u_e.

To construct a *MPG* with respect to a source, the vertices and the edges can be easily obtained from a given *FRG* by breadth-first-search. To determine the edge

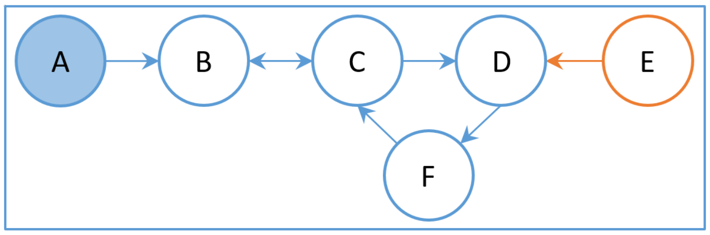


Figure 6.6: An example *MPG*. The entire graph is a *FRG* consisting of 6 vertices. The vertex A is the source. A, B, C, D, F and the edges between them form the *MPG* with respect to A. Since there is no directed path from A to E, vertex E is not included in the *MPG*.

weights, we propose a method based on *graph signal processing* as described in **Algorithm 6.1**. The motivation behind this method is that, for a *FR* $u_e \rightarrow u_r$ and a message m issued by u_e, the more semantically similar are the message histories of u_e and u_r, and the more semantically similar are m and the message history of u_r, the more likely m is to be propagated by u_r (assuming that u_r sees m). For example, suppose u_e's message history is primarily about the topic *machine learning*, and u_e now issues a message m. If m is about a new deep learning model, and u_r has a big interest in *machine learning* as well, then it is very likely that u_r will propagate m after seeing it.

Algorithm 6.1:

Given:

- A message m.
- A subgraph $\mathcal{G}_{u_s} = (\mathcal{V}_{u_s}, \mathcal{E}_{u_s})$ of the *FR* graph of an *SSNM* induced by a given source vertex u_s (i.e. $\mathcal{G}_{u_s}$ contains u_s and all of its *in-neighbors* (vertices that have an edge from u_s to the vertex), and all edges induced by these vertices).
- A message history (within a specified range) D_i for the user u_i associated to each vertex v_i of $\mathcal{G}_{u_s}$.
- A finite set T of *tau*'s (*tau* > 0) for the Heat kernel bank.

- A *logistic function* [33] $\varphi_p : \mathbb{R} \to [0, 1]$.

Seek:

A weight assignment to the edges of $\mathcal{G}_{u_s}$ thereby making it a *MPG*.

Construct Graph Signal:

FOR $v_i \in \mathcal{V}_{u_s}$:

Compute the semantic similarity $Sim_d(m, D_i)$ between m and D_i by utilizing *DSCTP*;

A graph signal is defined as a function: $s_m : v_i \mapsto Sim_d(m, D_i)$;

Weight Assignment:

Compute a weight assignment to $\mathcal{G}_{u_s}$ by solving the following optimization problem (using **Algorithm 6.2**):

Minimize $\sup\limits_{v_i \in \mathcal{V}_{u_s}} |s_m(v_i) - \hat{s}_{\tilde{\mathcal{G}}_{u_s}}(v_i)|$, subject to the initial condition: $\tilde{\mathcal{G}}_{u_s} = \mathcal{G}_{u_s}$, where $\tilde{\mathcal{G}}_{u_s}$ denotes the updated $\mathcal{G}_{u_s}$ with a new weight assignment, and $\hat{s}_{\tilde{\mathcal{G}}_{u_s}}$ denotes the filtered signal computed by applying the Heat kernel bank to s_m to do the low-pass filtering on $\tilde{\mathcal{G}}_{u_s}$ similar to the signal filtering in **Algorithm 4.1**;

LET $w_{\tilde{\mathcal{G}}_{u_s}} : \mathcal{E}_{u_s} \to \mathbb{R}$ denote a solution to this optimization problem.

Probability Conversion:

FOR $e_{jk} \in \mathcal{E}_{u_s}$, where e_{jk} is the edge from v_j to v_k:

Convert $w_{\tilde{\mathcal{G}}_{u_s}}(e_{jk})$ to a probability by computing $\varphi_p(w_{\tilde{\mathcal{G}}_{u_s}}(e_{jk}))$.

RETURN those resulting probabilities.

Algorithm Discussion:

We make several notes about **Algorithm 6.1**.

First, when collecting the message history for the users, a time range needs to be specified, and every user should use the same range.

Second, the logistic function φ_p has the form: $\varphi_p = \frac{Max}{1+e^{-k(x=x_0)}}$, where Max is the maximum value that φ_p can output, $k > 0$ controls the curvature of φ_p, x_0 is the midpoint of φ_p such that $\varphi_p(x_0) = \frac{Max}{2}$. Some example logistic functions are shown in Figure 6.7. To make φ_p adaptive to **Algorithm 6.1**, Max should be set to 1. Here, x_0 is *not* necessarily 0, because $w_{\tilde{\mathcal{G}}_{u_s}}(e_{jk}) = 0$ implies that u_j and u_k do not have semantically similar message histories that are semantically similar to m, and in this case, intuitively the likelihood that u_k propagates m when seeing m from u_j should be very low rather than "half-and-half".

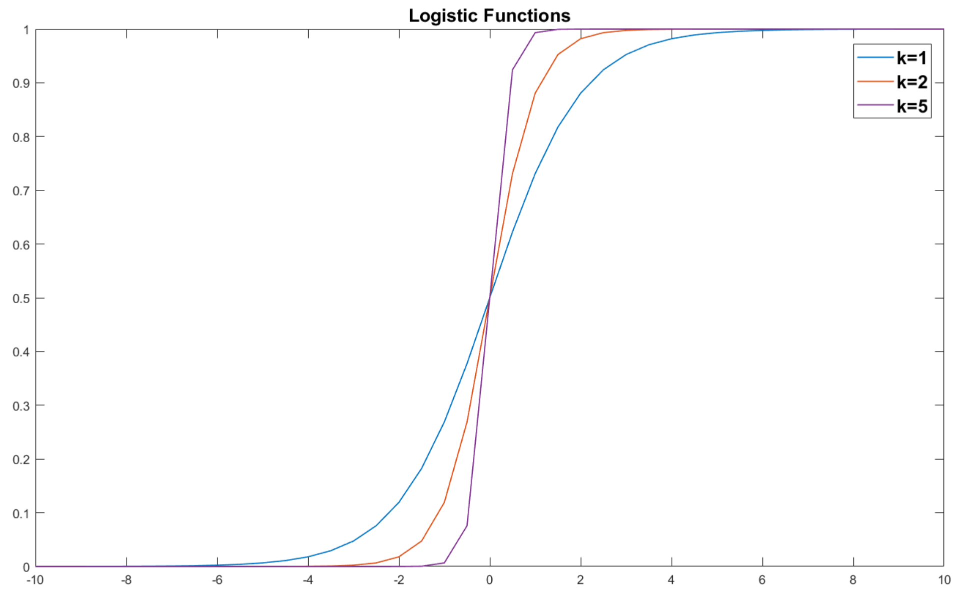


Figure 6.7: Examples of logistic functions. The larger the k, the steeper the curve.

Third, as discussed in Chapter 4, computing filtered signals requires the use of the graph Fourier transform which is defined based on a Laplacian on graphs. However, the graphs considered in **Algorithm 6.1** are directed graphs instead of undirected graphs as considered in Chapter 4. Then what Laplacian to use is an issue. Some previous work has discussed Laplacians on directed graphs and the relevant issue on the graph Fourier transform. Chung et al. proposed in [28] a Laplacian for directed graphs motivated by random walks; however, it requires the irreducibility (i.e. strongly connected) of the graphs. In [139], an asymmetric Laplacian on directed graphs is defined by using *in-degrees* [104] of vertices. In [88], an asymmetric Laplacian on directed graphs is proposed, and it also requires the irreducibility. In [130], instead of directly proposing a Laplacian, the authors discuss how to build the graph Fourier basis from an optimization problem perspective formulated by using *Lovász extension* [91]. Ideally, a symmetric Laplacian without requiring irreducibility will perfectly fit into our problem. And, in fact, the unnormalized version of Chung's Laplacian proposed in [28] satisfies these properties; however, the authors of [88] point out that the normalized version of Chung's Laplacian could not effectively capture the unique characteristics of random walks on directed graphs because different random walks may lead to the same Laplacian. This issue also exists in the unnormalized version of Chung's Laplacian. This point can be seen from the *Teoplitz decomposition* [166]. The unnormalized version of Chung's Laplacian can be expressed as $L_s = \frac{1}{2}(L + L^T)$ where $L = D - A$ is the Laplacian matrix of a directed graph, A is the corresponding adjacency matrix and D is the degree matrix induced from A. By the *Teoplitz decomposition*, there exists a counterpart of L_s, which is $L_a = \frac{1}{2}(L - L^T)$, and also, there exists $L_d = L_s + L_a$ (which is the decomposition form of a matrix L_d). In this decomposition, L_s reflects the symmetric component of L_d and L_a reflects the asymmetric component. Intuitively, L_d is also a Laplacian, which encodes both the symmetric characteristics and the asymmetric characteristics of a directed graph. Thus, L_d is more sophisticated than Chung's Laplacian. However, L_d is not symmetric. On the other hand, the authors of [139] show that the Laplacian proposed in their work (which actually is L) does have some important properties that are friendly to the graph Fourier transform such as 0 being guaranteed to be an eigenvalue and real parts of the eigenvalues being non-negative for a positive-weighted graph. Naturally, those properties also hold in L_s. To date, choosing an ideal Laplacian on directed graphs for the graph Fourier transform is still an open problem. As a trade-off solution, we use the unnormalized version of Chung's Laplacian.

Fourth, to use **Algorithm 6.1**, an optimization problem has to be solved: finding a $\tilde{\mathcal{G}}_{u_s}$ to minimize $\sup_{v_i \in \mathcal{V}_{u_s}} |s_m(v_i) - \hat{s}_{\tilde{\mathcal{G}}_{u_s}}(v_i)|$. To attack this problem, we propose an algorithmic framework in **Algorithm 6.2**:

Algorithm 6.2:

Given:

- A directed and weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$.
- A signal $s : \mathcal{V} \to \mathbb{R}$ defined on $\mathcal{G}$.
- A finite set T of *tau*'s ($tau > 0$) for the Heat kernel bank.
- A threshold $\theta_{gd} > 0$.
- A weight update function φ_w that takes a vertex v_k as input and updates the weights on the edges incident to v_k.

Seek:

A directed and weighted graph $\tilde{\mathcal{G}}$ minimizing $\sup_{v_i \in \mathcal{V}} |s(v_i) - \hat{s}_{\tilde{\mathcal{G}}}(v_i)|$, where $\hat{s}_{\tilde{\mathcal{G}}}$ is the filtered signal by Heat kernel with T.

```
Initially, $\tilde{\mathcal{G}} = \mathcal{G}$.
***START***:
    **LET** $\delta_{gd} = \sup_{v_i \in \mathcal{V}} |s(v_i) - \hat{s}_{\tilde{\mathcal{G}}}(v_i)|$ for the current $\tilde{\mathcal{G}}$.
    **LET** $\nabla s = \{|s(v_1) - \hat{s}_{\tilde{\mathcal{G}}}(v_1)|, \ldots, |s(v_N) - \hat{s}_{\tilde{\mathcal{G}}}(v_N)|\}$ denote the collection of absolute differences between $s(v_i)$ and $\hat{s}_{\tilde{\mathcal{G}}}(v_i)$ indexed by $v_i$;
    Sort $\nabla s$ in descending order;
***UPDATE WEIGHTS***:
    Mark all elements in $\nabla s$ as *available*;
    **IF** All elements in $\nabla s$ are marked as *unavailable*:
        **RETURN** $\tilde{\mathcal{G}}$;
    **ELSE**:
        Fetch the top element from $\nabla s$ that is marked as *available* (indexed at $k$) from $\nabla s$;
        Compute $\varphi_w(v_k, \mathcal{W}_{v_k})$ to update the weights of edges incident to $v_k$;
        **LET** $\hat{\mathcal{G}}$ denote the updated $\mathcal{G}$ with $\hat{\mathcal{W}}_{v_k}$;
        Compute $\tilde{\delta}_{gd} = \sup_{v_i \in \mathcal{V}} |s(v_i) - \hat{s}_{\hat{\mathcal{G}}}(v_i)|$ with updated $\hat{\mathcal{G}}$;
    **IF** $\delta_{gd} > \tilde{\delta}_{gd}$:
        **LET** $\tilde{\mathcal{G}} = \hat{\mathcal{G}}$;
        **GOTO** ***START***;
    **ELSE IF** $\delta_{gd} < \theta_{gd}$:
        **RETURN** $\tilde{\mathcal{G}}$;
    **ELSE**:
        Mark the currently fetched element (indexed at $k$) of $\nabla s$ as *unavailable*;
        **GOTO** ***UPDATE WEIGHTS***;
```

Note that the weight update function φ_w can be implemented in multiple ways. The goal in designing this function is that for each iteration, φ_w should at most only slightly modify the weights so as to hopefully not miss a local optima. The choice of the threshold θ_{gd} will affect the running time of this algorithm. If θ_{gd} is small, then it may take many more iterations to find a solution than it does with a large value. Designing φ_w and determining θ_{gd} will be considered in our future work.

By utilizing **Algorithm 6.1**, we can obtain a *MPG* for a message m and a source user on a *SSNM*. The weight on each edge is the probability of the *follower* propagating a message seen from a *followee*. To compute the probability of a user in the *SSNM* propagating m, we use the following algorithm:

Algorithm 6.3:

Given:

- A *MPG* $\mathcal{G}_{MPG} = (\mathcal{V}_{MPG}, \mathcal{E}_{MPG})$ with respect to a message m initially issued by a user u_s.

- A threshold θ_p for propagation probability.
- A user $u_i \in \mathcal{V}_{MPG}$.
- A *logistic function* $\varphi_q : \mathbb{R} \to [0,1]$ similar to φ_p used in **Algorithm 6.1** is used to convert a real value to a value bounded between 0 and 1 which can be taken as a probability.

Seek:

A probability of u_i propagating m.

Step 1:

Enumerate all paths (may not be *simple* paths) from u_s to u_i such that, for each path $\mathcal{P}_l(u_s, u_i) = (e_{sk}, \ldots, e_{hi})$, the product of probabilities on this path $\prod_{e_{xy} \in \mathcal{P}_l(u_s,u_i)} e_{xy} > \theta_p$;

Denote the set of all such paths by $\mathbb{P}(u_s, u_i) = \{\mathcal{P}_l(u_s, u_i)\}$;

Step 2:

Compute $\varphi_q\Big(\sum_{\mathcal{P}_l(u_s,u_i) \in \mathbb{P}(u_s,u_i)} \prod_{e_{xy} \in \mathcal{P}_l(u_s,u_i)} e_{xy}\Big)$ to obtain the desired probability.

Note that the threshold θ_p will affect the running time of **Step 1**. The lower the threshold, the more paths that will be found and the more time that will be required.

6.4 Conclusion

The work in this chapter will be continued as one of the primary Post-doc activities of the author by this dissertation. The social network model proposed in this chapter is a well-formalized platform that can be adapted to various studies about message propagations in social network environments. The methods proposed to address the *MP* problem form a significant vanguard exploration in interdisciplinary studies across social dynamics and natural language processing.

Chapter 7
FUTURE WORK

Document semantic representation is a profound topic, and to date it is far from being well-developed. Exploring this topic from diverse perspectives is significant. Our works, *DSCTP*, *APV* and *GPGS*, have initiated a new direction in the study of the *DSR* problem, namely, utilizing *GPs*, algebraic topology and *GSP* techniques. Attesting to the competitive performance of our methods on various tasks from the experimental results, we are confident about the future of this direction, and there are remaining open problems. In this chapter, we briefly summarize some important future work in three aspects: first, work to improve *DSCTP*, *APV* and *GPGS*; second, soft document clustering; and third, social dynamics problems. These three aspects are detailed in Sections 7.1, 7.2 and 7.3.

7.1 Future Works for DSCTP, APV and GPGS

In this section, we discuss some key future work related to *DSCTP*, *APV* and *GPGS*.

7.1.1 DSCTP

Here, we list five featured future works for *DSCTP*.

The first future work is to determine the lexical similarity threshold required by *DSCTP*. Although, from our experimental results, *DSCTP* is not sensitive to this threshold, it seems important to provide a more rigorous and effective approach to set the value. The key difficulty in determining the threshold is that when *DSCTP* is applied to different datasets the threshold that leads to the best performance may vary significantly. For example, on *Lee60* and *Li30*, *DSCTP* works the best with 0.6 and

0.5 respectively, while on *20News-M5* 0.3 is the best choice. Studying the underlying reasons causing such differences is a key step in solving this threshold determination issue.

The second future work is about computing minimum cycle bases. Recall that *DSCTP* requires the computation of minimum basic cycles so that semantically similar *GPs* can be extracted. As discussed in Section 2.4, generally a minimum cycle basis will not be unique in a graph. However, seeking all minimum cycle bases can take an exponential running time. This motivates a significant future work to investigate trade-off solutions that can capture a large portion of semantically similar *GPs* meanwhile keeping the running time small (preferentially polynomial).

The third future work is to deal with some difficult cases when using *GPs*. For example, consider the sentence: "*A dinosaur flees when it sees Errol*", and its *CBPT* shown in Figure 7.1. In the *CBPT*, "*dinosaur flees*" is a *GP*, and it has a common-sense meaning. The path length of this *GP* is 4 (i.e. a short path) which indicates that this *GP* is significant. "*dinosaur ... sees*" is also a *GP*, and obviously it is as meaningful as "*dinosaur flees*". However, its path length is significantly longer than 4 even after pruning, which weakens its significance. This happens because "*it*" is a pronoun and will be removed by the pruning. Then even though actually "*it sees*" should be the *GP* that carries the meaning of "*dinosaur sees*" and is significant, what will be extracted from the sentence is "*dinosaur ... sees*".

The fourth future work is to improve the performance of *DSCTP* in sentence semantics comparisons. At the end of Chapter 2, we exemplify a couple of cases of sentence semantics comparison in which *DSCTP* may not perform well. We explain that there are two major reasons. One is that there may not be an adequate number of semantically similar *GPs* to contribute to the similarity of two sentences that are short. The other is that constituency relations used to define *GPs* may not be effective in capturing dependency relations. Thus, considering sentence lengths and dependency relations may give improvements to *DSCTP*.

The fifth future work is to normalize the similarity values produced by *DSCTP*.

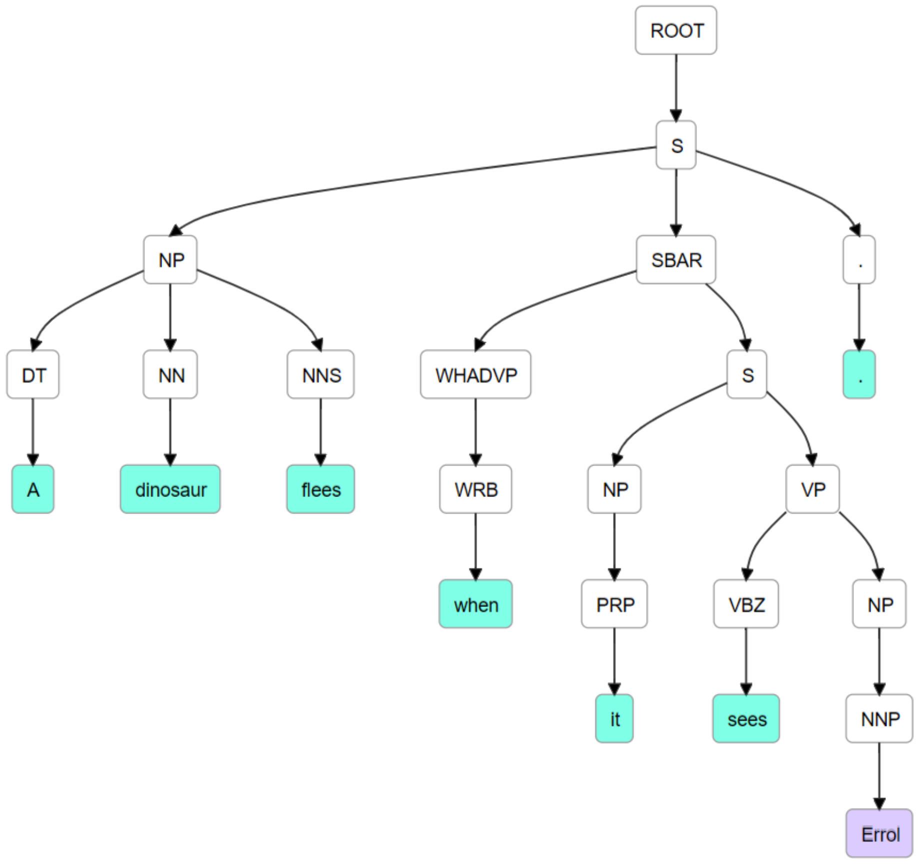


Figure 7.1: The *CBPT* of "*A dinosaur flees when it sees Errol*".

In our current design, *DSCTP* does not provide normalized values, which leads to difficulties in determining if a similarity value is great or not. A similarity value has to be compared with other values so that we can understand what it really reflects. This is not convenient in practical use. However, *DSCTP* computes similarity values by using a weighted sum which produces resulting values that are proportional to the number of minimum basic cycles. Thus, the similarity values are not bounded from above. This may cause difficulties in the normalization. Future work is needed to address this issue and find a normalization method.

7.1.2 APV & GPGS

The first future work for *APV* and *GPGS* is studying how to make an appropriate background document set. In Chapter 3, we propose several design guidelines for selecting background documents. Nevertheless, in complex scenarios such as social network problems, appropriately selecting background documents is difficult. For example, in our design guidelines, the background documents should contain topics of interest. However, topics may not be given when selecting background documents. In addition, since topics typically have hierarchical structures, and at different levels topics are more abstract or more specific, then selecting topics effectively can be difficult. Moreover, it is very likely that human efforts may be needed in the background document selection and the topic selection. If so, then how to make the human work as easy as possible is also a significant future work.

The second future work is to seek alternative approaches to cosine similarity to compute similarities or distances of *APVs* and *GPGS's*. We mentioned in Chapter 5 that *Bregman* distances [17] may be used. However, there are obstacles when using them. For example, to utilize *Jensen-Shannon divergence* [32], the vector values of *APVs* and *GPGS's* are required to be probabilities. Then how to convert vector values to probabilities is a non-trivial problem. In addition, how *APVs* and *GPGS's* would benefit from using alternative similarity and distance methods is an interesting question.

The third future work is specific to *GPGS*. As shown in Section 5.6, *GPGS* may have overestimated induced significance values. The reason is that at each vertex the induced values will be accumulated and the accumulation is not bounded from above. To address this issue, an approach better than accumulation to compute the significance value based on induced values is needed. Intuitively, from the heat diffusion point of view, for example, when we heat up two spots on a steel sheet at the same time, the temperature at the middle point between the two hot spots cannot be higher than the temperature at each hot spot. Thus, the temperature at the middle point is not simply an accumulation of the temperatures at this point caused by the diffusion from the two hot spots respectively. This intuition motivates a future work to design a better approach to utilize induced values.

The fourth future work is about the dimensionality reduction for *GPGS*. In

Section 5.5, we have shown that the number of non-zero dimensions of *GPGS* can be higher than 1000, and in some cases it can be higher than 5000. Such high dimensional spaces may encounter the *curse of dimensionality* issue [65]. The key problem is that many of these dimensions reflect similar semantics because each dimension corresponds to a single *GP*. This is an intrinsic issue in the design of *GPGS* where there is no *GP* clustering so as to avoid the confused clustering issue. Thus, a dimensionality reduction is needed to remove the dependencies caused by semantic similarities between dimensions. This dimensionality reduction may not be linear because the dependencies may not be linear. Non-linear dimensionality reduction (NLDR) itself is an area for development. Designing an effective and efficient NLDR method to address the issue deserves a future work.

7.2 Soft Document Clustering

In this dissertation, we utilize *DSCTP* to address the *DSC* problem, and utilize *DSCTP*, *APV* and *GPGS* to address the *HDC* problem. Another significant future work is to explore other applications to which these methods can be applied. An immediate application successive to the *HDC* problem is the **soft document clustering (SDC)** problem. The *SDC* problem takes a set of documents as input, and seeks a soft clustering of these documents with respect to document semantics. Here, a document can be placed in multiple clusters. For example, consider a document discussing artificial intelligence marketing analysis. Intuitively, this document is very likely to contain both a semantics centered on artificial intelligence and a semantics centered on marketing. Then in a *SDC* problem instance such a document should be clustered to both a group in which artificial intelligence is its major semantics and another group in which marketing is its major semantics.

The key to solving the *SDC* problem is to find semantic overlaps between documents. Thus, studying if *APVs* and *GPGS's* can be effective in representing semantic overlaps is a future work. Specifically, we believe that, similar to *Word2Vec*, algebraic operations, such as $+$ and $-$, also hold and are meaningful on *APVs* and *GPGS's*. In future work, we will examine if this is true. If we could find all significant semantic overlaps in a set of documents, then each such overlap will correspond to a cluster. One difficulty in this approach is that we need to measure the significance of the semantics in an overlap and in each document. For example, both a document that just barely mentions artificial intelligence and a document that heavily details artificial intelligence techniques have a semantics about artificial intelligence, but the significances of the semantics in the two documents are majorly different, and reasonably they should not be clustered into the same group (where artificial intelligence is the primary semantics). Thus, a future work will be needed to solve this problem.

7.3 Social Dynamics Problems

The research on the *MP* problem discussed in Chapter 6 will be continued as one of the primary Post-doc activities of the author by this dissertation. To address the *MP* problem, solving the optimization problem proposed in **Algorithm 6.1** is the key, and as a potential solution **Algorithm 6.2** will be implemented and tested. Another important consideration in solving the *MP* problem is the time and space efficiency. The data comes from real social network crawls (which is huge). Thus, the time and space efficiency of *DSCTP*, *APV* and *GPGS* are significant concerns. For this reason, optimizing these methods is a future work.

7.4 Conclusion

Designing more effective *DSRs* and exploring additional applications of these *DSRs* are our goals. In this dissertation, we specifically study document which is a concrete object that carries semantics. However, semantics is an abstract object. Studying other concrete objects (e.g. images) that can carry semantics, and comparing semantics carried by different concrete objects, will be an important future work.

BIBLIOGRAPHY

[1] Omri Abend and Ari Rappoport. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 228–238, 2013.

[2] Omri Abend and Ari Rappoport. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 77–89, 2017.

[3] Ameya Agaskar and Yue M Lu. An uncertainty principle for functions defined on graphs. In *Wavelets and Sparsity XIV*, volume 8138, page 81380T. International Society for Optics and Photonics, 2011.

[4] Ameya Agaskar and Yue M Lu. A spectral graph uncertainty principle. *IEEE Transactions on Information Theory*, 59(7):4338–4356, 2013.

[5] Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics, 2012.

[6] Stephan Arndt, Carolyn Turvey, and Nancy C Andreasen. Correlating and predicting psychiatric symptom ratings: Spearmans r versus kendalls tau correlation. *Journal of psychiatric research*, 33(2):97–104, 1999.

[7] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Unsupervised statistical machine translation. *arXiv preprint arXiv:1809.01272*, 2018.

[8] Valentina Emilia Balas, Sanjiban Sekhar Roy, Dharmendra Sharma, and Pijush Samui. *Handbook of Deep Learning Applications*, volume 136. Springer International Publishing, 2019.

[9] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, 2013.

[10] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[11] Nicole Berline, Ezra Getzler, and Michele Vergne. *Heat Kernels and Dirac Operators.* Springer Science & Business Media, 2003.

[12] Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. Bracketing guidelines for treebank ii style penn treebank project. *University of Pennsylvania*, 97:100, 1995.

[13] David M Blei and John D Lafferty. Topic models. *Text Mining: Classification, Clustering, and Applications*, 10(71):34, 2009.

[14] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[15] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[16] Harold Borko and Myrna Bernick. Automatic document classification. *Journal of the ACM (JACM)*, 10(2):151–162, 1963.

[17] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

[18] Joan Bresnan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. *Lexical-Functional Syntax*, volume 16. John Wiley & Sons, 2015.

[19] Chris Buckley and Gerard Salton. *Onix Text Retrieval Toolkit Stopword List 2*, 2007.

[20] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. Nasari: A novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, 2015.

[21] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64, 2016.

[22] John Rozier Cannon. *The One-Dimensional Heat Equation*, volume 23. Cambridge University Press, 1984.

[23] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[24] Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. Importance of semantic representation: Dataless classification. In *Aaai*, volume 2, pages 830–835, 2008.

[25] Walter G Charles. Contextual correlates of meaning. *Applied Psycholinguistics*, 21(4):505–524, 2000.

[26] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.

[27] Noam Chomsky. *Syntactic Structures*. Walter de Gruyter, 2002.

[28] Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, 2005.

[29] Ronald R Coifman, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the national academy of sciences*, 102(21):7426–7431, 2005.

[30] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[31] Richard Courant and David Hilbert. *Methods of Mathematical Physics: Partial Differential Equations*. John Wiley & Sons, 2008.

[32] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[33] Jan Salomon Cramer. The origins of logistic regression. 2002.

[34] Common Crawl. Common crawl. *URl: http://http://commoncrawl. org*.

[35] Alan Cruse. Meaning in language: An introduction to semantics and pragmatics. 2011.

[36] E Brian Davies, Josef Leydold, and Peter F Stadler. Discrete nodal domain theorems. *arXiv preprint math/0009120*, 2000.

[37] Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970.

[38] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[39] James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, 2007.

[40] Paul Adrien Maurice Dirac. *The Principles of Quantum Mechanics*. Number 27. Oxford university press, 1981.

[41] Steven N Durlauf and H Peyton Young. *Social Dynamics*, volume 4. Mit Press, 2004.

[42] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Soc., 2010.

[43] Katrin Erk. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653, 2012.

[44] Yue Feng, Ebrahim Bagheri, Faezeh Ensan, and Jelena Jovanovic. The state of the art in semantic relatedness: A framework for comparison. *The Knowledge Engineering Review*, 32, 2017.

[45] John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

[46] Edward B Fowlkes and Colin L Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983.

[47] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems (TOIS)*, 9(3):223–248, 1991.

[48] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2018.

[49] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.

[50] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[51] Wael H Gomaa and Aly A Fahmy. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18, 2013.

[52] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM, 1961.

[53] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd international conference on Machine learning*, pages 377–384. ACM, 2006.

[54] David J Griffiths. Introduction to electrodynamics, 2005.

[55] Alexander Grigoryan. *Heat Kernel and Analysis on Manifolds*, volume 47. American Mathematical Soc., 2009.

[56] Lushan Han, Abhay L Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. Umbc_ebiquity-core: semantic textual similarity systems. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, volume 1, pages 44–52, 2013.

[57] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[58] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[59] Allen Hatcher. Algebraic topology. 2002. *Cambridge UP, Cambridge*, 606(9), 2002.

[60] Werner Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. In *Original Scientific Papers Wissenschaftliche Originalarbeiten*, pages 478–504. Springer, 1985.

[61] Geoffrey Hinton. *2015 Speech to the Royal Society in London.*

[62] Roger A Horn and Charles R Johnson. *Matrix Analysis.* Cambridge university press, 2012.

[63] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

[64] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[65] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[66] Ray Jackendoff. *Semantic structures*, volume 18. MIT press, 1992.

[67] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[68] Dan Jurafsky and James H Martin. *Speech and Language Processing*, volume 3. Pearson London, 2014.

[69] David E Kanouse and L Reid Hanson Jr. Negativity in evaluations. In *Preparation of this paper grew out of a workshop on attribution theory held at University of California, Los Angeles, Aug 1969.* Lawrence Erlbaum Associates, Inc, 1987.

[70] Telikepalli Kavitha, Christian Liebchen, Kurt Mehlhorn, Dimitrios Michail, Romeo Rizzi, Torsten Ueckerdt, and Katharina A Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199–243, 2009.

[71] Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna E Paluch. An õ (m^2n) algorithm for minimum cycle basis of graphs. 2004.

[72] Earle H Kennard. Zur quantenmechanik einfacher bewegungstypen. *Zeitschrift für Physik*, 44(4-5):326–352, 1927.

[73] David J Ketchen and Christopher L Shook. The application of cluster analysis in strategic management research: An analysis and critique. *Strategic Management Journal*, 17(6):441–458, 1996.

[74] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[75] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015.

[76] Solomon Kullback. *Information Theory and Statistics.* Courier Corporation, 1997.

[77] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *ICML*, 2015.

[78] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

[79] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[80] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[81] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[82] Michael D Lee, Brandon Pincombe, and Matthew Welsh. An empirical evaluation of models of text document similarity. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 27, 2005.

[83] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[84] David Lewis et al. Reuters-21578. *Test Collections*, 1, 1987.

[85] Mike Lewis and Mark Steedman. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192, 2013.

[86] Hang Li. Deep learning for natural language processing: advantages and challenges. *National Science Review*, 2017.

[87] Yang Li and Tao Yang. Word embedding for understanding natural language: A survey. In *Guide to Big Data Applications*, pages 83–104. Springer, 2018.

[88] Yanhua Li and Zhi-Li Zhang. Digraph laplacian and the degree of asymmetry. *Internet Mathematics*, 8(4):381–401, 2012.

[89] Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Rngineering*, 18(8):1138–1150, 2006.

[90] Sebastian Löbner. *Understanding Semantics*. Routledge, 2013.

[91] László Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

[92] Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

[93] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[94] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[95] Igor Melcuk. Levels of dependency in linguistic description: Concepts and problems. *Dependency and Valency. An International Handbook of Contemporary Research*, 1:188–229, 2003.

[96] Peter Mika. Ontologies are us: A unified model of social networks and semantics. In *International semantic web conference*, pages 522–536. Springer, 2005.

[97] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[98] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[99] George A Miller and Walter G Charles. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28, 1991.

[100] James R Munkres. *Elements of Algebraic Topology*, volume 4586. Addison-Wesley Longman, 1984.

[101] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[102] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[103] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[104] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

[105] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2002.

[106] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[107] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

[108] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[109] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet:: Similarity: Measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics, 2004.

[110] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[111] Alessandro Perina, Nebojsa Jojic, Manuele Bicego, and Andrzej Truski. Documents as multiple overlapping windows into grids of counts. In *Advances in Neural Information Processing Systems*, pages 10–18, 2013.

[112] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. Rt to win! predicting message propagation in twitter. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[113] Jonathan Pfautz. *Computational Simulation of Online Social Behavior (SocialSim)*, 2017.

[114] Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1341–1351, 2013.

[115] Manfred Pinkal. *Logic and Lexicon: The Semantics of the Indefinite*, volume 56. Springer Science & Business Media, 2013.

[116] Martin Potthast and Benno Stein. New issues in near-duplicate detection. In *Data Analysis, Machine Learning and Applications*, pages 601–609. Springer, 2008.

[117] David MW Powers. Applications and explanations of zipf's law. In *Proceedings of the joint conferences on new methods in language processing and computational natural language learning*, pages 151–160. Association for Computational Linguistics, 1998.

[118] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning*, pages 147–155. Association for Computational Linguistics, 2009.

[119] R Rehurek and P Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2), 2011.

[120] Tom Richardson and Ruediger Urbanke. *Modern Coding Theory.* Cambridge university press, 2008.

[121] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[122] Bernard Rosner and Robert J Glynn. Interval estimation for rank correlation coefficients based on the probit transformation with extension to measurement error correction of correlated ranked data. *Statistics in medicine*, 26(3):633–646, 2007.

[123] Herbert Rubenstein and John B Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[124] Walter Rudin. *Real and Complex Analysis.* Tata McGraw-hill education, 2006.

[125] John Saeed. I. 2003. semantics. *GB: Blackwell Publishing*, 1997.

[126] Alexandre Salle, Marco Idiart, and Aline Villavicencio. Matrix factorization using window sampling and negative sampling for improved word representations. *arXiv preprint arXiv:1606.00819*, 2016.

[127] Mark Sanderson. Duplicate detection in the reuters collection. *" Technical Report (TR-1997-5) of the Department of Computing Science at the University of Glasgow G12 8QQ, UK"*, 1997.

[128] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE Transactions on Signal Processing*, 61(7):1644–1656, 2013.

[129] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014.

[130] Stefania Sardellitti, Sergio Barbarossa, and Paolo Di Lorenzo. On the graph fourier transform for directed graphs. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):796–811, 2017.

[131] Adriaan MJ Schakel and Benjamin J Wilson. Measuring word significance using distributed representations of words. *arXiv preprint arXiv:1508.02297*, 2015.

[132] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to Information Retrieval*, volume 39. Cambridge University Press, 2008.

[133] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

[134] David J Sheskin. *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2003.

[135] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *arXiv preprint arXiv:1211.0053*, 2012.

[136] Grigori Sidorov. Syntactic dependency based n-grams in rule based automatic english as second language grammar correction. *International Journal of Computational Linguistics and Applications*, 4(2):169–188, 2013.

[137] Grigori Sidorov, Francisco Velasquez, Efstathios Stamatatos, Alexander Gelbukh, and Liliana Chanona-Hernández. Syntactic dependency-based n-grams as classification features. In *Mexican International Conference on Artificial Intelligence*, pages 1–11. Springer, 2012.

[138] Grigori Sidorov, Francisco Velasquez, Efstathios Stamatatos, Alexander Gelbukh, and Liliana Chanona-Hernández. Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3):853–860, 2014.

[139] Rahul Singh, Abhishek Chakraborty, and BS Manoj. Graph fourier transform based on directed laplacian. In *2016 International Conference on Signal Processing and Communications (SPCOM)*, pages 1–5. IEEE, 2016.

[140] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1201–1211. Association for Computational Linguistics, 2012.

[141] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[142] John F Sowa. Semantic networks. 1987.

[143] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[144] Mark Steedman. *The syntactic process*, volume 24. MIT press Cambridge, MA, 2000.

[145] Michael Steinbach, George Karypis, Vipin Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

[146] Catherine A Sugar and Gareth M James. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association*, 98(463):750–763, 2003.

[147] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[148] TERENCE TAO. Fourier transform. 2015.

[149] Lucien Tesnière. Eléments de syntaxe structurale. 1959.

[150] Steven K. Thompson. *Sampling*. John Wiley & Sons, 2012.

[151] Lloyd N Trefethen. *Spectral Methods in MATLAB*, volume 10. Siam, 2000.

[152] Mircea Trifan, Bogdan Ionescu, Cristian Gadea, and Dan Ionescu. A graph digital signal processing method for semantic analysis. In *2015 IEEE 10th Jubilee International Symposium on Applied Computational Intelligence and Informatics*, pages 187–192. IEEE, 2015.

[153] Mikhail Tsitsvero, Sergio Barbarossa, and Paolo Di Lorenzo. Signals on graphs: Uncertainty principle and sampling. *IEEE Transactions on Signal Processing*, 64(18):4845–4860, 2016.

[154] Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[155] William T Vetterling, Saul A Teukolsky, William H Press, and Brian P Flannery. *Numerical Recipes: The Art of Scientific Computing.*, volume 2. Cambridge university press Cambridge, 1992.

[156] Philippe Vismara. Union of all the minimum cycle bases of a graph. *the electronic journal of combinatorics*, 4(1):9, 1997.

[157] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[158] Peter Weiner. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 1–11. IEEE, 1973.

[159] Arnold D Well and Jerome L Myers. *Research design & statistical analysis.* Psychology Press, 2003.

[160] Hermann Weyl. Quantenmechanik und gruppentheorie. *Zeitschrift für Physik*, 46(1-2):1–46, 1927.

[161] Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, 2016.

[162] Anna Wierzbicka. *The semantics of grammar*, volume 18. John Benjamins Publishing, 1988.

[163] William A Woods and R Kaplan. Lunar rocks in natural english: Explorations in natural language question answering. *Linguistic structures processing*, 5:521–569, 1977.

[164] Weichao Xu, Yunhe Hou, YS Hung, and Yuexian Zou. Comparison of spearman's rho and kendall's tau in normal and contaminated normal models. *arXiv preprint arXiv:1011.2009*, 2010.

[165] Jun Yan and Jian Hu. *Text Semantic Representation*, pages 1–4. Springer New York, New York, NY, 2016.

[166] Ke Ye and Lek-Heng Lim. Every matrix is a product of toeplitz matrices. *Foundations of Computational Mathematics*, 16(3):577–598, 2016.

[167] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *SIGIR*, volume 98, pages 46–54. Citeseer, 1998.

[168] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M Karp. Fast and intuitive clustering of web documents. In *KDD*, volume 97, pages 287–290, 1997.

[169] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2005.

[170] Fan Zhang and Edwin R Hancock. Graph spectral image smoothing using the heat kernel. *Pattern Recognition*, 41(11):3328–3342, 2008.

[171] Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V Vasilakos. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017.

[172] Donald W Zimmerman, Bruno D Zumbo, and Richard H Williams. Bias in estimation and hypothesis testing of correlation. *Psicológica*, 24(1), 2003.

Appendix A

GRAPH FOURIER TRANSFORMATION: AN EXPLANATION

A Fourier transform on graphs is an analogue to the *Fourier transform* in *harmonic analysis*. To obtain such a Fourier transform, we take advantage of eigenvectors of the combinatorial *Laplacian* on graphs. Here we explain why this analogue is reasonable.

To appreciate this analogue, we consider a specific class of functions. For any *Lebesgue measurable* [124] function $u : \mathbb{R} \rightarrow \mathbb{C}$[1] satisfying that $u \in L^2$ and $u \in L^1$ and $u \in BV(\mathbb{R})$[2], where

$$L^p = \Big\{ u : \big(\int_{-\infty}^{\infty} |u(x)|^p \big)^{\frac{1}{p}} < \infty \Big\} \tag{A.1}$$

and

$$BV(\mathbb{R}) = \Big\{ u : ||u||_{TV} = \sup_{\mathcal{X}=\{x_0<\cdots<x_n\}\subset\mathbb{R}} \sum_{i=0}^{n} |u(x_{i+1}) - u(x_i)| < \infty, \, \mathcal{X} \text{ is finite} \Big\} \tag{A.2}$$

the function u can be represented as the *superposition* of a set of *plane waves* in the form of $A(x, 0) = A_0 e^{i2\pi x\xi}$, where $\xi \in \mathbb{R}$ is known as the *frequency* of the plane wave [3].

[1] For convenience, we only consider single-variable functions here. Note that multi-variable functions will provide similar cases.

[2] $BV(\mathbb{R})$ is the space of functions of *bounded variation*. Every function in this class has bounded *total variation*.

[3] Formally, a *plane wave* is defined by $A(x, t) = A_0 \cos(kx - \omega t + \varphi)$, where A_0 is a complex number whose magnitude is the *amplitude* of the wave, k is the *wave number* which is equal to $\frac{2\pi}{\lambda}$ where λ is the wavelength, x is the spatial position, ω is the *angular frequency* which is equal to $\frac{2\pi}{T}$ where T is the wave period, t is the time point, and φ is the *phrase shift*. Equivalently, in the exponential form, $A(x, 0) = A_0 e^{i(kx-\omega t+\varphi)}$. In our case, it is not necessary to consider how a plane wave varies along with time or phrase shifts. Thus, for convenience, we set t and φ to zero, and thereby simplify the exponential form to $A(x, 0) = A_0 e^{ikx}$. Conventionally, the wave number k is also known as the

More specifically, the superposition is expressed as

$$u(x) = \int_{-\infty}^{\infty} \hat{u}(\xi) e^{i2\pi x\xi} d\xi \tag{A.3}$$

where

$$\hat{u}(\xi) = (\mathcal{F}u)(\xi) = \int_{-\infty}^{\infty} e^{-i2\pi x\xi} u(x) dx, \quad \xi \in \mathbb{R}. \tag{A.4}$$

Intuitively, $\hat{u}(\xi)$ represents "how much" of the plane wave at the frequency ξ is contained in u, and also $\hat{u}(\xi)$ can be regarded as an inner product of u and $B(x) = e^{i2\pi x\xi}$ **Equation** A.4 gives the definition of the *Fourier transform*, and **Equation** A.3 is the *Fourier inversion theorem.*

The *Fourier transform* is tightly related to the *Laplacian* operator [148]. The Laplacian operator is defined by

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}. \tag{A.5}$$

Thus, for a plane wave function $B(x) = e^{i2\pi x\xi}$,

$$\Delta B(x) = -4\pi^2\xi^2 e^{i2\pi x\xi} = (-4\pi^2\xi^2)B(x). \tag{A.6}$$

This implies that $B(x)$ is a *generalized eigenfunction*[4] for the Laplacian operator, and $-4\pi^2\xi^2$ is the corresponding *eigenvalue*. Also, it is well known that the *Laplacian matrix* on graphs can be interpreted as a particular case of the *discrete Laplacian operator* which is an analogue to the Laplacian operator shown above. Then, naturally, *eigenvectors* and *eigenvalues* of the Laplacian matrix can be interpreted as an analogue to eigenfunctions and eigenvalues of the Laplacian operator. Therefore, it is reasonable to make an analogue as follows: for a Laplacian matrix defined over a combinatorial

spatial frequency of a wave, which roughly speaking measures the number of waves per unit distance. When the context of *spatial* is clarified, we use the term *frequency* for short. The readers interested in details about plane waves should consult the books [54] and [40].

[4] Plane waves are not square integrable.

graph, by computing the inner product between a function defined on the vertices and the eigenvectors of the Laplacian matrix, the result is an analogue to the Fourier transform of the function. Specifically, the **graph Fourier transform** is defined as follows:

Def: Graph Fourier Transform:

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected, weighted and undirected graph with $|\mathcal{V}| = N$, and let L be the Laplacian matrix $(N \times N)$ for $\mathcal{G}$. Also, let $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ be the eigenvalues of L, and let $\{g_1, g_2, \ldots, g_N\}$ be the corresponding eigenvectors. Then, for a function $f : \mathcal{V} \to \mathbb{R}$ defined on $\mathcal{G}$, the **graph Fourier transform** of f is a vector, and each vector value is defined by [135]:

$$\hat{f}(\lambda_i) = \langle \boldsymbol{f}, g_i \rangle = \sum_{k=1}^{N} f(k) g_i^*(k) \tag{A.7}$$

where $\boldsymbol{f}$ is the vector form of f and $f(k)$ is the function value for the kth vertex. λ_i's are called the **frequencies** with respect to the graph Fourier transform.

The **inverse graph Fourier transform** is defined by:

$$f(i) = \sum_{k=1}^{N} \hat{f}(\lambda_k) g_k(i) \tag{A.8}$$

In this definition, since eigenvalues of the Laplacian matrix are playing the roles of frequencies, then it is also expected that the eigenvectors corresponding to the higher eigenvalues oscillate more rapidly, and the eigenvectors corresponding to the lower eigenvalues oscillate more slowly. Several studies [106] [135] [4] have provided experimental evidence for this property. More specifically, these studies witnessed that in some cases eigenvectors corresponding to higher eigenvalues have more zero-crossings than eigenvectors corresponding to lower eigenvalues. However, it can be found that though from a qualitative point of view eigenvectors do have this property, in some cases, some eigenvectors may violate it. A counter example is shown in Figure 4.4 and A.1. In Figure A.1, $0 = \lambda_1 \leq \cdots \leq \lambda_5$ are the eigenvalues, and the colored lines are the corresponding eigenvectors. The green line shows the eigenvector corresponding to the highest eigenvalue, and it has two zero-crossings. The yellow line corresponds to the third eigenvalue, and it has three zero-crossings. These two eigenvectors violate the property. On the other hand, *Courant's nodal domain theorem* [31] has been generalized to discrete graphs (called the **discrete nodal domain theorem**) [36], which supports the property necessarily (yet not sufficiently). In [36], given a function defined on the vertices of a graph, a **strong positive (negative) sign graph** with respect to the function is defined as a maximal connected subgraph on which the function is positive (negative); and a **weak positive (negative) sign graph** is defined as a maximal connected subgraph on which the function is non-negative (non-positive) with at least one function value being positive (negative). The *discrete nodal domain theorem* states that any eigenvector corresponding to the kth eigenvalue (in ascending algebraic order) has at most $k + r - 1$ strong sign graphs and at most k weak sign graphs, where r is the multiplicity of kth eigenvalue. However, it is still far from a proof for this property. Thus, none of the intuitive understanding from the analogue to the classic Fourier transform, the experimental evidence, or the *discrete nodal domain theorem* provide a convincing explanation of this property. To make applications of *GSP* rigorous, future work is needed to study exactly when this property would hold or not, and thereby how the graph Fourier transform would be impacted.

To further justify the analogue, we would like to examine one of the most important properties of the Fourier transform: the smoothness of u implying the decay of $\hat{u}$[5]. Specifically, the Fourier transforms of smooth functions would decay rapidly to zero at infinity. Intuitively, this property implies that the smoother the function, the fewer the high-frequency components in its Fourier transform. This property is well

[5] Similarly, the decay of u implies the smoothness of $\hat{u}$.

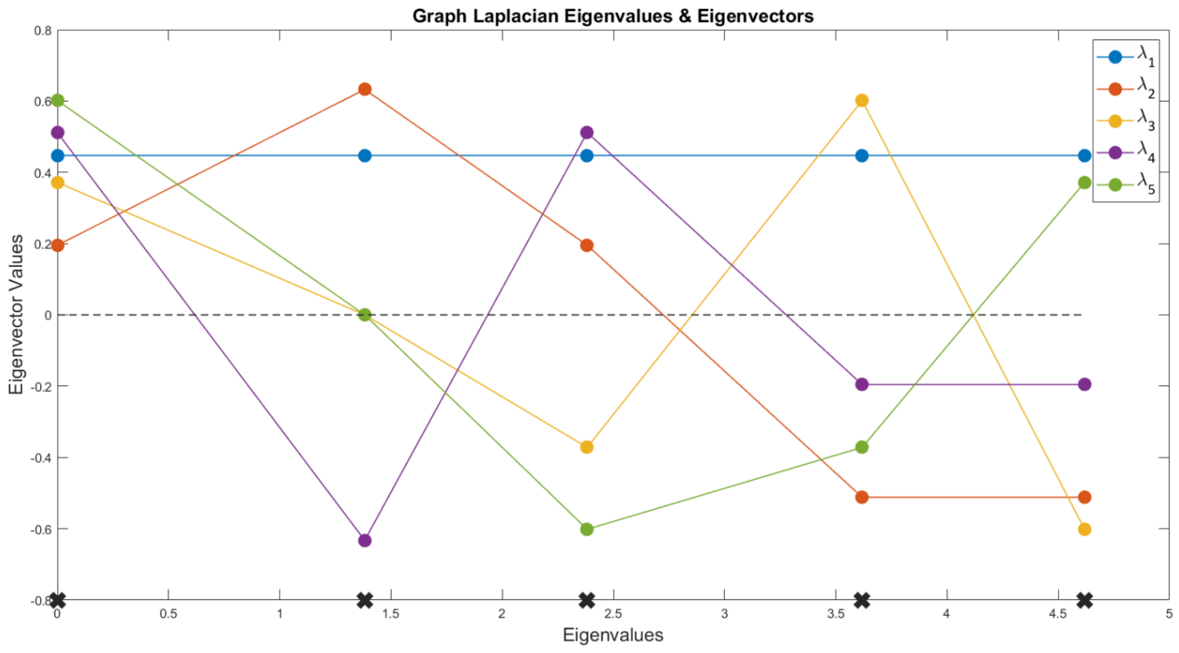


Figure A.1: Eigenvalues and eigenvectors of the Laplacian matrix induced by the graph shown in Figure 4.4. Here, $0 = \lambda_1 \leq \cdots \leq \lambda_5$ are eigenvalues, and are marked with a black "X" at the bottom. The eigenvectors are shown with different colors corresponding to the eigenvalues colored in the legend.

described in Trefethen's book [151] *Theorem 1*. Particularly, it is straightforward[6] to obtain a proposition: if a function $u \in BV(\mathbb{R})$, then $\hat{u}(\xi) = O(|\xi|^{-1})$. This proposition is not difficult to verify. If we assume that u has an order 1 continuous derivative (i.e. $u \in C^1$), then we have

$$i2\pi\xi\hat{u}(\xi) = \int_{-\infty}^{\infty} e^{-i2\pi x\xi} u'(x)dx + e^{-i2\pi x\xi} u(x) \tag{A.9}$$

By taking modulus on both sides, since $u \in BV(\mathbb{R})$[7], then we have

$$|2\pi\xi||\hat{u}(\xi)| \leq \int_{-\infty}^{\infty} |u'(x)|dx = ||u||_{TV} < \infty \tag{A.10}$$

[6] In Trefethen's book [151], *Theorem 1* has four statements. The first statement is that: if u has $p-1$ continuous derivatives in L^2 for some $p \geq 0$ and a pth derivative of bounded variation, then $\hat{u}(k) = O(|k|^{-p-1})$ as $|k| \to \infty$. Our proposition is simply a special case of this statement.

[7] If $u \in C^1$ and $u \in BR(\mathbb{R})$, then $\int_{-\infty}^{\infty} |u'(x)|dx < \infty$.

For the case $u \notin C^1$, the total variation of u, instead of $||u||_{TV} = \int_{-\infty}^{\infty} |u'(x)|dx$, is expressed as $||u||_{TV} = \sup_{\mathcal{X}=\{x_0<\cdots<x_n\}\subset\mathbb{R}} \sum_{i=0}^{n} |u(x_{i+1}) - u(x_i)|$, and thus the proposition still holds.

On the other hand, in the graph case, if we use the eigenvectors of the Laplacian matrix to compute the Fourier transform of a given function defined on a graph, we expect the Fourier transform on graphs to have the same smoothness-decay property. Here, we show that it does. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a weighted and undirected graph, and let $f : \mathcal{V} \to \mathbb{R}$ be a function defined on $\mathcal{G}$. Then the total variation of f is defined by the order 2 *p-Dirichlet-form* [135]:

$$||f||_{TV} = S^2(f) = \sum_{(i,j)\in\mathcal{E}} w_{ij}[f(j) - f(i)]^2 \tag{A.11}$$

Moreover, $S^2(f)$ coincides with the *quadratic form* of the Laplacian matrix L (i.e. $S^2(f) = \boldsymbol{f}^T L\boldsymbol{f}$). Then intuitively, there must be an intrinsic relationship between the total variation $S^2(f)$ and the *spectrum* (i.e. the eigenvalues and the eigenvectors) of the Laplacian matrix. This relationship, according to the *Courant-Fischer theorem* [62][8], can be described as a series of optimization problems which are stated in a recursive form [135]:

$$\lambda_0 = \min_{\substack{\boldsymbol{f}\in\mathbb{R}^N \\ ||f||_2=1}} \boldsymbol{f}^T L\boldsymbol{f} \tag{A.12}$$

$$\lambda_k = \min_{\substack{\boldsymbol{f}\in\mathbb{R}^N \\ ||f||_2=1 \\ \boldsymbol{f}\perp span(g_0,\ldots,g_{k-1})}} \boldsymbol{f}^T L\boldsymbol{f}, \qquad k > 0 \tag{A.13}$$

where λ_i's are eigenvalues of L, and g_i's are eigenvectors of L. Each of these optimization problems can be solved by utilizing the *Rayleigh theorem* [62]. This implies that if f is a minimizer for the ith optimization problem, then f can be selected as the

[8] The Laplacian matrix is *Hermitian* (i.e. it is identical to its *conjugate transpose*), so the *Courant-Fischer theorem* is applicable.

ith eigenvector of L. Then it is straightforward to show the relationship between the graph Fourier transform and the total variation of a function defined on the graph in the following proposition (we use the same settings as in the definition of the graph Fourier transform above).

Proposition 4.1:

$$\frac{\langle \boldsymbol{f}, g_i \rangle}{|\boldsymbol{f}|} \le \frac{||f||_{TV}}{\lambda_i}, \qquad \text{for } \lambda_i \neq 0. \tag{A.14}$$

Proof:

As discussed above where g_i is the minimizer for minimizing $\boldsymbol{f}^T L \boldsymbol{f}$ to compute λ_i, then for a function f:

$$||f||_{TV} = \boldsymbol{f}^T L \boldsymbol{f} \ge \langle L g_i, g_i \rangle = \langle \lambda_i g_i, g_i \rangle = \lambda_i \langle g_i, g_i \rangle \tag{A.15}$$

Since $\langle \boldsymbol{f}, g_i \rangle = |\boldsymbol{f}| \langle \boldsymbol{f}_{\mathbb{1}}, g_i \rangle$, then:

$$\frac{\langle \boldsymbol{f}, g_i \rangle}{|\boldsymbol{f}|} = \langle \boldsymbol{f}_{\mathbb{1}}, g_i \rangle \le \langle g_i, g_i \rangle \le \frac{||f||_{TV}}{\lambda_i} \quad \square \tag{A.16}$$

Note that when $\lambda_i = 0$ the corresponding eigenvector is a constant vector, and thus the minimum possible total variation for a function is 0 (0 is achieved when the function is a constant function). The proposition indicates that for a *frequency* λ_i (i.e. an eigenvalue) the closer (with respect to the inner product) the function f and the eigenvector g_i, the more the Fourier transform is taking the component at this frequency; moreover, the smoother the function f, the more the low-frequency components and the less the high-frequency components are taken. Therefore, the graph Fourier transform does preserve the smoothness-decay property.

Appendix B

DATASET MANIFEST

B.1 20Newsgroups

The *20Newsgroups* dataset is provided by [78][1]. The two subsets, *20News-C10* and *20News-M5*, of this dataset used in our experiments are listed in Sections B.1.1 and B.1.2.

B.1.1 20News-C10

alt.atheism

51141, 51184, 51186, 51199, 51201, 51230, 51233, 51279, 51284, 51311, 53056, 53068, 53093, 53095, 53130, 53132, 53198, 53218, 53221, 53226, 53282, 53284, 53297, 53303, 53309, 53319, 53329, 53337, 53406, 53454, 53455, 53465, 53467, 53486, 53503, 53522, 53571, 53572, 53574, 53579, 53588, 53635, 53640, 53784, 54129, 54160, 54172, 54220, 54262, 54471.

comp.sys.ibm.pc.hardware

58924, 58927, 58978, 60192, 60220, 60232, 60263, 60378, 60388, 60408, 60415, 60443, 60476, 60483, 60497, 60529, 60532, 60582, 60687, 60688, 60697, 60698, 60719, 60737, 60739, 60779, 60780, 60794, 60837, 60908, 60914, 60919, 60925, 60934, 60967, 61010, 61021, 61037, 61048, 61056, 61057, 61064, 61069, 61072, 61078, 61087, 61094, 61128, 61149, 61177.

misc.forsale

74737, 74747, 74754, 74766, 74814, 75852, 75863, 75864, 75876, 75884, 75922, 75953, 75959, 75965, 75975, 75976, 75996, 76006, 76010, 76019, 76020, 76026, 76028, 76029,

[1] The original dataset can be downloaded from: http://qwone.com/ jason/20Newsgroups/.

76038, 76098, 76103, 76120, 76128, 76175, 76220, 76245, 76268, 76295, 76302, 76318, 76335, 76461, 76478, 76489, 76577, 76581, 76585, 76592, 76593, 76594, 76624, 76648, 76783, 76824.

rec.motorcycles

103118, 103128, 103139, 103159, 103165, 103201, 103207, 103211, 104356, 104360, 104395, 104398, 104404, 104408, 104414, 104427, 104432, 104482, 104520, 104596, 104635, 104640, 104652, 104672, 104694, 104699, 104732, 104734, 104737, 104754, 104778, 104797, 104800, 104821, 104829, 104949, 104967, 104975, 104989, 105069, 105104, 105112, 105122, 105155, 105203, 105231, 105237, 105243, 105380, 105564.

rec.sport.hockey

52580, 52595, 52628, 52639, 52643, 53537, 53539, 53544, 53599, 53609, 53628, 53654, 53706, 53734, 53735, 53737, 53745, 53765, 53783, 53791, 53799, 53800, 53864, 53873, 53888, 53895, 53919, 53969, 53971, 53974, 53982, 54008, 54013, 54060, 54071, 54073, 54104, 54194, 54240, 54243, 54252, 54291, 54293, 54295, 54296, 54510, 54724, 54748, 54766, 54778.

sci.electronics

52733, 52774, 52820, 52831, 53513, 53514, 53521, 53524, 53529, 53534, 53580, 53588, 53608, 53610, 53627, 53665, 53672, 53705, 53737, 53752, 53770, 53793, 53796, 53798, 53827, 53828, 53833, 53910, 53912, 53958, 53984, 53991, 54020, 54075, 54102, 54150, 54159, 54169, 54176, 54185, 54202, 54248, 54251, 54262, 54271, 54274, 54277, 54334, 54338, 54495.

sci.med

58057, 58066, 58074, 58116, 58128, 58142, 58814, 58816, 58828, 58835, 58842, 58848, 58885, 58919, 58923, 58955, 58961, 58976, 58987, 59018, 59023, 59026, 59047, 59053, 59075, 59096, 59119, 59124, 59144, 59146, 59157, 59200, 59211, 59227, 59262, 59281, 59294, 59313, 59336, 59364, 59366, 59425, 59448, 59471, 59481, 59486, 59525, 59532, 59624, 59630.

sci.space

60159, 60162, 60178, 60211, 60224, 60782, 60795, 60813, 60822, 60827, 60845, 60886, 60898, 60973, 60978, 60983, 60993, 61037, 61047, 61075, 61080, 61098, 61103, 61107, 61113, 61171, 61178, 61192, 61197, 61202, 61218, 61240, 61259, 61288, 61303, 61312, 61343, 61349, 61355, 61441, 61450, 61466, 61488, 61492, 61501, 61514, 61517, 61549, 61555, 62479.

talk.politics.guns

53311, 53326, 53332, 53336, 53338, 53370, 54155, 54177, 54200, 54213, 54236, 54250, 54252, 54280, 54281, 54301, 54366, 54374, 54402, 54426, 54452, 54462, 54472, 54481, 54542, 54562, 54584, 54633, 54643, 54735, 54749, 54753, 54837, 54877, 54885, 54887, 54889, 54906, 54908, 54959, 55075, 55078, 55081, 55093, 55107, 55124, 55270, 55426, 55468, 55486.

talk.politics.mideast

75911, 75922, 75950, 75967, 75977, 75996, 76004, 76039, 76052, 76062, 76065, 76085, 76098, 76105, 76106, 76111, 76141, 76194, 76212, 76215, 76227, 76250, 76255, 76287, 76349, 76382, 76384, 76408, 76433, 76439, 76441, 76445, 76451, 76480, 76482, 76569, 77179, 77210, 77231, 77241, 77252, 77256, 77262, 77290, 77302, 77321, 77324, 77353, 77376, 77399.

B.1.2 20News-M5

From *20News-C10*, we select five categories to form *20News-M5*. These five categories are:

- **comp.sys.ibm.pc.hardware**
- **rec.motorcycles**
- **rec.sport.hockey**
- **sci.space**
- **talk.politics.mideast**

B.2 Reuters-21578

The *Reuters-21578* dataset is provided by [84][2]. The subset, *Reuters-M7*, of this dataset used in our experiments is listed in Section B.2.1. In addition, the categories in *Reuters-21578* that contains more than 50 documents are listed in Section B.2.2.

B.2.1 Reuters-M7

acq

1076, 10867, 10918, 11101, 11386, 11505, 11822, 11888, 12532, 14504, 16233, 16402, 16677, 16979, 17347, 17479, 17513, 17538, 17551, 17701, 17782, 18576, 19831, 20039, 21319, 21550, 2219, 2247, 3039, 3387, 3775, 3795, 4622, 4733, 5293, 5379, 5739, 6085, 6361, 6433, 6819, 7558, 7635, 7693, 7810, 840, 8518, 9087, 9509, 9833.

crude

10011, 11455, 12286, 12609, 12940, 13184, 1343, 14732, 15322, 15520, 16077, 17093, 17100, 17161, 17408, 17816, 18066, 18401, 18523, 18651, 19560, 19903, 19927, 19998, 20778, 20878, 21443, 21492, 242, 3174, 3303, 4569, 4713, 489, 5123, 5125, 5281, 5630, 6598, 6746, 7067, 7097, 7496, 7611, 7684, 8209, 8835, 8959, 9849, 988.

gold

10485, 1072, 1082, 10868, 11627, 13757, 14842, 15803, 15811, 1607, 16286, 16971, 17163, 17264, 17622, 18388, 19764, 19802, 19808, 20699, 2073, 21314, 21576, 2411, 2762, 2782, 2785, 2880, 2983, 314, 3322, 3327, 3625, 488, 5209, 5481, 5526, 5541, 5558, 5564, 5958, 6872, 7023, 7166, 7985, 8757, 8857, 8877, 8948, 9104.

interest

11244, 11314, 12398, 12406, 12453, 12469, 12627, 12731, 12774, 13144, 13231, 13262, 1427, 15410, 15816, 16383, 16951, 17071, 17204, 17222, 17448, 17943, 18520, 18564, 18904, 19201, 19512, 1976, 1995, 20532, 20769, 21342, 233, 2601, 3002, 3122, 411, 4458, 4720, 6350, 6359, 6369, 6483, 6543, 7460, 7614, 8153, 8168, 8184, 8802.

livestock

[2] The original dataset can be downloaded from: http://www.daviddlewis.com/resources/testcollections/reuters21578/.

10616, 121, 12332, 12376, 12594, 12595, 12618, 13382, 1372, 1393, 1398, 15237, 15290, 15949, 16236, 16316, 16519, 17090, 17165, 17399, 17424, 18373, 18676, 19269, 19555, 20620, 2448, 2495, 2594, 2851, 2978, 3278, 3318, 3339, 3340, 3901, 4022, 443, 4503, 4514, 4833, 5009, 5031, 5400, 7924, 7962, 8574, 8744, 9290, 9585.

ship

10620, 12823, 13074, 14818, 14957, 16076, 16962, 17073, 17117, 17403, 17412, 18372, 18426, 18851, 18865, 19055, 19092, 19122, 19238, 20103, 21526, 2853, 2944, 2955, 2959, 3017, 3028, 3329, 3349, 3473, 3572, 3796, 3955, 4051, 4711, 4739, 4740, 4778, 518, 5330, 5734, 6436, 6652, 7119, 8214, 8246, 857, 8950, 9314, 9356.

soybean

10172, 10705, 10882, 11739, 11939, 11984, 12348, 12711, 12757, 13915, 1394, 1405, 14706, 15686, 15865, 15953, 15999, 16525, 16752, 17337, 18177, 18408, 18409, 18417, 18427, 18544, 1882, 19057, 20645, 229, 2382, 297, 303, 3282, 3453, 3458, 3467, 4076, 4083, 4599, 5531, 5702, 6116, 6890, 69, 6906, 7356, 7700, 8140, 8522.

B.2.2 36 Categories with More Than 50 Documents

livestock, gold, yen, trade, alum, dlr, sugar, rice, coffee, earn, nat-gas, money-fx, wheat, crude, jobs, carcass, corn, bop, acq, grain, copper, ipi, cocoa, soybean, interest, meal-feed, gas, gnp, iron-steel, reserves, ship, cotton, veg-oil, oilseed, cpi, money-supply.

B.3 BBC

The *BBC* dataset is provided by [53][3]. The subset, *BBC-M5*, of this dataset used in our experiments is listed in Section B.3.1.

B.3.1 BBC-M5

business

[3] The original dataset can be downloaded from: http://mlg.ucd.ie/datasets/bbc.html

011, 018, 038, 060, 096, 108, 109, 123, 124, 129, 144, 158, 169, 194, 233, 234, 237, 255, 276, 278, 280, 283, 301, 321, 322, 343, 352, 355, 369, 372, 373, 377, 385, 394, 396, 408, 410, 418, 431, 432, 433, 463, 470, 473, 480, 483, 492, 496, 500, 510.

entertainment

001, 003, 005, 019, 060, 065, 083, 087, 089, 092, 107, 112, 139, 144, 154, 157, 160, 165, 170, 174, 177, 186, 198, 200, 209, 211, 222, 230, 232, 233, 242, 243, 249, 260, 266, 270, 272, 274, 290, 302, 309, 310, 317, 318, 320, 332, 343, 344, 359, 380.

politics

004, 010, 015, 017, 025, 026, 028, 035, 037, 043, 044, 054, 057, 074, 077, 091, 108, 123, 129, 140, 144, 145, 150, 163, 171, 178, 188, 191, 206, 212, 235, 236, 237, 243, 249, 269, 291, 304, 308, 309, 310, 329, 330, 343, 381, 391, 397, 398, 412, 413.

sport

006, 007, 015, 019, 029, 031, 052, 056, 074, 077, 083, 084, 085, 092, 098, 116, 130, 136, 143, 162, 167, 171, 173, 175, 191, 194, 203, 220, 229, 239, 249, 256, 261, 262, 279, 317, 319, 443, 444, 454, 457, 467, 471, 485, 498, 501, 504, 505, 506, 508.

tech

015, 024, 030, 036, 054, 057, 060, 070, 080, 111, 117, 162, 176, 181, 182, 207, 209, 232, 233, 237, 238, 239, 241, 242, 247, 248, 252, 254, 257, 258, 262, 271, 272, 284, 303, 314, 317, 327, 330, 331, 333, 334, 339, 341, 344, 345, 352, 374, 377, 380.