VECTOR REPRESENTATION OF DOCUMENTS USING WORD CLUSTERS

Sunanda Bansal

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

September 2021

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Sunanda Bansal**
Entitled: **Vector Representation of Documents using Word Clusters**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________ Examiner
(Dr. Gregory Butler)

______________________________ Examiner
(Dr. Leila Kosseim)

______________________________ Thesis Supervisor
(Dr. Sabine Bergler)

Approved by ______________________________
(Dr. Lata Narayanan)
Chair of Department

____
(Dr. Mourad Debbabi)
Dean,
Gina Cody School of Engineering and Computer Science

Abstract

Vector Representation of Documents using Word Clusters

Sunanda Bansal

For processing the textual data using statistical methods like Machine Learning (ML), the data often needs to be represented in the form of a vector. With the dawn of the internet, the amount of textual data has exploded, and, partly owing to its size, most of this data is unlabeled. Therefore, often for sorting and analyzing text documents, the documents have to be represented in an *unsupervised* way, i.e. with no prior knowledge of expected output or labels. Most of the existing unsupervised methodologies do not factor in the similarity between words, and if they do, it can be further improved upon. This thesis discusses Word Cluster based Document Embedding (WcDe) where the documents are represented in terms of clusters of similar words and, compares its performance in representing documents at two levels of topical similarity - general and specific. This thesis shows that WcDe outperforms existing unsupervised representation methodologies at both levels of topical similarity. Furthermore, this thesis analyzes variations of WcDe with respect to its components and discusses the combination of components that consistently performs well across both topical levels. Finally, this thesis analyses the document vector generated by WcDe on two fronts, i.e. whether it captures the similarity of documents within a class, and whether it captures the dissimilarity of documents belonging to different classes. The analysis shows that Word Cluster based Document Embedding is able to encode both aspects of document representation very well and on both of the topical levels.

Acknowledgments

In every endeavour of my life, I've been supported by many, probably too many. This thesis is no different. I would like to take this moment to thank the people who have supported me in various phases of this journey.

First and foremost, I would like to thank Dr. Sabine Bergler. Sabine, you've taught me how to communicate. I've learnt not only to express with more clarity but also to interpret with more objectivity. From you, I've learnt as much about the professional conduct as a researcher as the research methodology itself. Above all, for your immense patience and professionalism during my personal struggles, I can not thank you enough. I really appreciate your support and training throughout my studies. This thesis would not be the same without it. Thank you!

When I started, Artificial Intelligence was popular but I wasn't sure if I'd like it. For changing my mind, and setting me on a path of no return, I want to thank Dr. Leila Kosseim. Your course on Artificial Intelligence and my work with you, later, as a Teaching Assistant for the course, led to further development of an interest in machine learning and deep learning methods. Without your step-by-step explanation of well-designed simple and thorough examples, the math would have been too daunting and I probably wouldn't have dared to get close to the realm of neural networks or word embeddings. Working through those examples as a student, and then as your TA, led to the development of intuition and an unyielding fascination with Artificial Neural Networks (ANNs). I owe the foundation and the development of my interest in the field to you. I was inspired.

One individual, in particular, has been my constant partner for all discussions with respect to my research and the development of my thesis - Aman Kumar. In the middle of a pandemic, from the other side of the globe, you spent the last 10 months with me, syncing your schedule with mine to help me manage the execution of my research and thesis. You've reveled in the results and you've stressed with me on the blocks. I loved how we built on top of each other's ideas for the analysis. For my endless discussions regarding every facet of this research, from the whiteboard discussions to reading my drafts, you've done it all. You are my brother and a peer, and your input and company were, and remain, invaluable in the formation of this thesis.

As it turns out I've had multiple partners for discussing my ideas and research. Here's to the ones that I bugged the most. Himani Saini, who's been my friend of 10 years, was my roommate for nearly 6, but who's also a most valuable fellow professional - you were my go-to person for all the discussion related to the research and random philosophical awanderings. Your input in research and your support in everything else was instrumental. Parsa Bagherzadeh, who is a friend and a cherished colleague from the ClaC Lab - I sincerely admire your readiness and enthusiasm to share your knowledge. We've had our intermittent discussions, and they have all been immensely helpful, always. Nadia Bilal, a dear friend, a fellow CLaC Lab member, and my constant companion while working on my thesis - your not being familiar with my research was the best asset for improving this thesis. You gave me your uninhibited opinion on the drafts, and for that, I am really grateful to you. Jerry George Thomas, who has been a friend, a roommate, and another fellow professional - I picked your brain more often than I dare to count. For letting me do so, every time, I thank you.

I would also like to thank Dataperformers. The realization of the problem and the need for research on the unsupervised representation of textual documents was a result of my work at Dataperformers. I am really grateful for the valuable experience, the excellent team and nourishing work environment that I got at Dataperformers. The entire team was very supportive, accommodating and understanding with respect to my studies. I'd like to thank Yoann Racine, in particular, who advised me with regards to publishing my code. I would also like to thank all the members of CLaC Lab. In particular, I would like to thank Narjes Tahei, a colleague in the ClaC lab. Your report on word representations and our discussion thereafter has been significant in shaping up my discussion for document representations. A shout out to all the ClaCers, especially, Parsa Bagherzadeh, Nadia Sheikh, and Nadia Bilal - it was fun because you all were there.

Most importantly, for their unwavering affection and patience through my personal struggles as well as my ramblings, I'd like to dedicate this milestone to my family. From every fibre of my being, I am grateful to my mother, my rock, my closest friend, who left no stone unturned to support me in every challenge I faced; to my father who gave me the push I needed, sometimes by doting and flattery, and sometimes by saying just the right thing at just the right time; to my brother who lost sleep (and his peace of mind) sorting my ramblings with me and my grandmother who repeatedly insisted I finished it asap because she missed me. Because there are many, too many - to more than I can mention, to all my friends and family, I thank you for being there.

And the last but definitely not the least, I'd like to thank Dr. Janet Ritchie. You saved my life. From the bottom of my heart, I thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In order to automate the analysis and processing of large amounts of data using statistical Artificial Intelligence (AI) methods, like Machine Learning (ML) models or Artificial Neural Networks (ANNs), the first thing to do with the data is to encode it in some mathematical form, usually in the form of real-valued vectors. This mathematical representation plays a key role in the performance of these methods as it serves as the input to the model and the starting point for automated statistical analysis. This mathematical representation stands in for the actual data and how well it encodes the essential features of the data plays a significant role in what the statistical AI methods can do with it. A simple statistical model using well-represented data as the input can usually stand shoulder to shoulder with a complex model that uses the same representation or worse. An excellent example of the importance of representation can be found in Computer Vision (CV). In Computer Vision, with the same underlying machine learning models, Convolutional Neural Network (CNN) layers were able to drastically improve the performance of CV systems by simply representing the image in such a dense real-valued vector that, as random as it may seem, captures essential features in the image. Such improvement in the representation of visual data and thereby, the results, established CNNs as the cornerstone of the success of Computer Vision. Therefore, it is vital how we represent the data and it is the representation of data, specifically textual data, that is at the center of the discussion in this thesis.

Data can usually be represented in a supervised or unsupervised fashion. If the expected outcome of the processing is indicated on a portion of data, i.e. a subset of the data is labeled, then this labeling can be leveraged to transform a basic initial representation of data to better reflect the expected outcome. This generates a new transformed representation, which is often dense and low-dimensional. Overall, the supervised representation of data allows for the data to be effectively represented according to particular labels. On the other hand, regardless of the availability of the labeled data, the data can be represented with no contribution or knowledge of the expected outcome (labels). Within the purview of unsupervised ways to represent data, there are generally two approaches that can be taken - task-specific representation and general. In the former, data is manually represented by features engineered to cater to a specific task whereas in the latter we attempt to automatically represent and capture essential features of data regardless of the task.

It is important to note here that there is a lot of data out there, often mostly unlabelled. Speaking of data, specifically of textual data, every day nearly 500 million tweets, 6 billion texts, and 10 billion Facebook messages are sent. It is a gross understatement to say that the amount of data generated in one internet day is huge. Much of the available is often unlabelled. On the other hand, data labeling is time-consuming and expensive, and therefore often not always a viable option. Therefore, for any analysis or to draw insights from such a large amount of unlabelled data, one simply cannot undermine the importance of unsupervised methods - be it sorting through your business product reviews to gather insights for improvement or sorting through a sudden influx of inquiries in the case of a worldwide pandemic. So with no knowledge of the labels, often the unsupervised methods are the only option one can resort to. To this end, even data has to be represented in an unsupervised way. However, it is difficult to manually analyze such large amounts of data. And it is even more difficult to manually engineer features that will ultimately effectively represent all of the data, especially based on a manual analysis of a very small part of it. Therefore, we need a way to automatically represent large amounts of data in a general and non-task-specific way. Therefore, out of the two approaches for unsupervised representation of data, this thesis will focus on the general, i.e. not task-specific, representation of data - specifically, textual data.

Furthermore, the role of representation is more pronounced when the labeled data is not available. For example, in the supervised statistical AI models, like Machine Learning models, the weights of the model are adjusted to produce results that align with the expected outcome. In these models, the representation of the data is ultimately weighted by the model, in some way. Therefore, to a certain degree, what the representation may lack, a supervised AI model can compensate by adjusting the weights. However, the unsupervised AI models operate without any knowledge of the expected output; there is no weighting of the representation of data to align with the expected output; the representation stands in for the data by itself and becomes the entire basis of the performance of the model. Therefore, for unsupervised processing of data, the initial representation of data is instrumental to the quality of the results. Since this thesis is focusing on discussing the representation of textual data, the representation will be evaluated in an unsupervised setting to compare the data representation in the absence of any knowledge of expected output. In other words, this thesis will address the unsupervised representation of textual data and evaluate it in an unsupervised data processing task, i.e. clustering.

Speaking of the general representation of textual data (i.e. not specific to a task), a good representation of a textual unit (e.g. a word, a sentence, etc.) is considered to be the one that best places the similar units closer in the vector space and dissimilar units further apart. In other words, a good representation of textual data captures its distinctive aspects and encodes it in the form of a vector. For the moment, let's focus on text documents that contain at least one sentence. Some techniques that have been popularly used to represent text documents in an unsupervised way completely bypass the inherent relatedness between different words of a language. However, with the advent of word embeddings in the last 2 decades or so, the words can now be represented in a mathematical form that encodes their contextual information and, based on this encoding, it places similar words closer in vector space and dissimilar ones further apart. Now, these word representations can be leveraged to represent text documents. However, many approaches that do use word embeddings to represent a text document, simply average the contextual information across each axis. This leads to a representation that completely loses the distinction between the words that was encoded in their vector representation. In other words, the similarity and dissimilarity encoded in the word representations are averaged out to somewhere in between and the distinction between the words that is encoded in the axes is not leveraged to its full potential. In this thesis, I will discuss a methodology that attempts to better utilize the contextual information encoded in word embeddings to represent textual documents. In this methodology, the word vectors obtained from word embeddings are clustered and these word clusters are then used to represent the documents. This underlying methodology, with slight variations, has been proposed and used by different researchers over the past 5-6 years. However, the discussion and research regarding this technique are scattered and are often lost in the heaps of publications. This thesis

- discusses the underlying methodology common to the multiple variations
- expands the variations of this methodology
- extends the evaluation to two topical levels
- compares this methodology with other unsupervised document representation methods
- compares various variations of this methodology with each other
- analyses the various components of the methodology
- draws insights that will provide a configuration of components that performs consistently well across different levels of topical similarity
- analyses the document vectors generated by the methodology
- attempts to consolidate much of the scattered research
- publishes a GitHub repository that implements the methodology and can be used to demo it. The repository is available at `https://github.com/sunandabansal/WcDe`.

Chapter 2

Unsupervised Representation of Texts - An Overview

Broadly speaking, in Natural Language Processing (NLP), the objective is automated understanding and/or generation of natural text or speech. In this thesis, the discussion is limited to the sub-domain of NLP that focuses on analyzing/understanding the language data present in textual form. Regardless of the medium, the natural language contains information at different unit levels - letter, word, phrase, sentence, paragraph, etc. The smallest unit of a language is a letter. A word is a unique sequence of these letters that expresses one or more concepts. At a point in time, the concepts associated with the word are more or less constant and universal. A phrase is a sequence of such words, which may or may not stand completely on its own to relay a complete thought, but forms a conceptual unit. A sentence is constituted of words or phrases within some grammatical structure of the language that expresses a complete stand-alone thought. One or more of such sentences can be considered a paragraph. Now that we've established that, in this thesis, I will use the term *document* to refer to any piece of text containing one or more sentences.

The textual data can be analyzed at any of the unit levels described above - word, alphabetical, sentence, etc. However, this thesis will particularly address the textual analysis at the document level. For analyzing the textual documents using statistical AI models, the text documents need to be represented in the form of a vector. In this thesis, our focus is the unsupervised representation of textual documents. This chapter will discuss the existing popular methodologies and underlying concepts related to the vector representation of documents. However, before we discuss vector representation of documents, let us briefly discuss vector spaces and the notation used hereafter.

A *vector* is any quantity defined with a direction and a magnitude. But for the linear algebra within the scope of this thesis, a vector can be thought of as a quantity that identifies a point in a Cartesian coordinate system. Such a vector forms a part of a *vector space*. A vector space is defined by a set of *coordinate axis*, or directions, which can then be used to define any vector that will identify any point in this space. The number of these coordinate axes in this vector space determines the *dimension* of the vector space. In this thesis, I will use Cartesian representation to

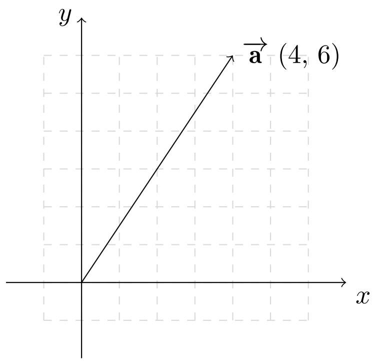


Figure 2.1: Example of a vector in a 2-dimensional vector space

represent the vectors. In this representation, the vector is broken down and represented in terms of its magnitude across the direction of each coordinate axis. The direction of each axes is indicated by a *unit vector* which is simply a vector of length 1. For example, let's consider a 2-dimensional vector $\overrightarrow{\mathbf{a}}$ that identifies a point with the coordinates $(4, 6)$ (Figure 2.1). This vector can be represented in terms of its magnitude in the direction of x and y axes respectively -

$$\overrightarrow{\mathbf{a}} = 4\hat{x} + 6\hat{y}$$

where, $\hat{x}$ and $\hat{y}$ are the unit vectors in the direction of x and y axes. A variable containing a vector is indicated with an arrow on top of the variable to differentiate between scalars and vectors. Similarly, a variable representing a unit vector is indicated with a cap/hat on the variable. In the field of statistical analysis in Artificial Intelligence, a vector is also understood as an array of real values (Goodfellow et al., 2016). In this array, each index of the array represents a feature of the data and the corresponding value, the weight of the feature in the data. Therefore, a vector representation of data is also known as a *feature vector* of the data, where each axis of the vector is referred to as a *feature* of the data. This term is particularly relevant for a particular class of approaches that can be taken to represent the textual data in NLP - feature engineering.

Among various unsupervised ways to represent text as a vector, one is that of manual feature engineering. In manual feature engineering, the data is analysed to determine statistical and/or linguistic aspects of the documents that are observed to be most relevant for the task at hand. Based on these features statistics are drawn leading to a real valued feature vector for the document. For example, consider a document d which contains one sentence - *I love reading.*. Let's say you have to represent it in terms of two features - the number of words and the number of characters in the document. Then this document can be represented in a feature vector as given below -

	Feature 1	Feature 2
	No. of words	No. of characters
	$\hat{x}$	$\hat{y}$
I love reading.	3	15

where $\hat{x}$ and $\hat{y}$ are the unit vectors representing the axis of feature 1 and 2. In Cartesian representation, the feature vector of the document d above, can be represented as

$$\overrightarrow{\mathbf{d}} = 3\hat{x} + 15\hat{y}$$

But as mentioned in the previous chapter, the feature-engineering-based approach can lead to document representations that specifically cater to a task. Moreover, this requires manual analysis of the data. However, this thesis focuses on general and automated unsupervised document representation techniques.

As mentioned in the Introduction, the goal here for a general vector representation of any entity - be it a word, a document, a text, or an image - is to map the similarity between the entities to the distance between their corresponding vectors in the vector space. That is, the relative distance between vectors should be indicative of the relative similarity between the two entities. Therefore, the goal of the vector representation of documents in this thesis is to represent documents in the form of a real-valued vector, in an unsupervised fashion, such that the closer the vectors, the similar the documents.

2.1 Document-Term Vector Space Model

One of the simplest ways to represent a document is to represent it in terms of the words it contains. In such representation of documents, the sequence of the words is disregarded and only the occurrence remains. In this thesis, such a model of document representation is referred to as Document-Term Vector Space Model (DT-VSM). Before we get into why it is referred to as the Document-Term Vector Space Model in this thesis, let's discuss what this model does. As I just said, in this model the document is represented in terms of the words it contains. To do so, a document is first split into word-level units called *tokens* and the process of splitting a text document into tokens is known as *tokenization*. Since it is a word-level unit in a text document it is also called a word in the text document. Unlike a word in a language that has a concept associated with it, a *word in a document* refers only to a word-level unit in the document. In this thesis, the term *word* is preferred to refer to word-level units because it is more relevant for the discussion in the thesis. When all the documents in the dataset are tokenized, the total set of words in the dataset is referred to as the *vocabulary* of the dataset and each element of the vocabulary is called a *term*. Therefore, to reiterate, in Document-Term Vector Space Model a document is represented in terms of the words it contains. For example -

Document 1	*they ran a 100m race and he ran a 500m race.*											
Words (Tokens)	*they*	*ran*	*a*	*100m*	*race*	*and*	*he*	*ran*	*a*	*500m*	*race*	*.*
Terms	*they*	*ran*	*a*	*100m*	*race*	*and*	*he*			*500m*		*.*

(Example 1.1)

Let's consider the document in the example above (Example 1.1). In Document-Term Vector Space Model, each document is represented as a vector in a vector space where each axis represents a term in the vocabulary of the dataset. But, the vocabulary of a dataset is usually big. Therefore, to fit the vector of our arbitrary example here, let's assume that the vocabulary of the dataset is limited to these 5 terms - *won*, *lost*, *ran*, *100m* and *race*. Given this vocabulary, let us represent the example document in terms of these 5 terms such that each of these 5 terms forms an axis of the document vector as shown below -

	won	*lost*	*ran*	*100m*	*race*
they ran a 100m race and he ran a 500m race.					

The empty cells above indicate the components of the document along each axis that are yet to be determined (hence, empty). This component will indicate an association between the term and the document in the form of importance, presence, or frequency of the term in that document. For the moment, let's represent the documents with a vector where the component indicates the frequency of each term (column), in the given document (row). Example 1.2 that follows shows the frequency of each term in the document from Example 1.1 as well as a few others.

			Axes				
			won	*lost*	*ran*	*100m*	*race*
			$\hat{t_1}$	$\hat{t_2}$	$\hat{t_3}$	$\hat{t_4}$	$\hat{t_5}$
d_1	:	they ran a 100m race and he ran a 500m race.	0	0	2	1	2
d_2	:	she won the 500m race but lost the rest.	1	1	0	0	1
d_3	:	he lost the 100m race and 200m race as well.	0	1	0	1	2

(Example 1.2)

where, the unit vector along i^{th} axis is represented as $\hat{t_i}$ and d_j refers to the j^{th} document. This mapping of terms and their frequencies in each document can be represented in the form of a matrix as given below -

$$\begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{bmatrix} 0 & 0 & 2 & 1 & 2 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 2 \end{bmatrix} \end{array}$$

(Example 1.3)

Such a matrix that indicates the occurrence of terms in documents is also known as an *document-term incidence matrix*. In the above matrix, each cell refers to the frequency of term (intersecting column) in the document (intersecting row). So, for example, the cell in the second row from the top (d_2) and the first column from the left (t_1) indicates that the term represented by t_1, i.e. *lost*, appears once in the document d_2, i.e. *she won the 500m race but lost the rest.* Such a matrix that gives the frequency of terms in a document is also called a *Document-Term Matrix*.

> A *Document-Term Matrix* is a document by term matrix which gives the frequency of each term in every document of the dataset. In a Document-Term Matrix, each row represents a document, each column represents a term and each cell at the intersection indicates the frequency of the term in the document.
>
> *(Definition 1)*

In a Document-Term Matrix, each row gives a vector that indicates the frequency of each term with the document. In other words, each row vector gives a vector representation of documents in Document-Term Vector Space Model. Based on the Document-Term Matrix (Example 1.3), the Cartesian representation of document vectors for each document from Example 1.2 is -

$$\begin{aligned}\overrightarrow{\mathbf{d_1}} &= 0\hat{t_1} + 0\hat{t_2} + 2\hat{t_3} + 1\hat{t_4} + 2\hat{t_5}\\ \overrightarrow{\mathbf{d_2}} &= 1\hat{t_1} + 1\hat{t_2} + 0\hat{t_3} + 0\hat{t_4} + 1\hat{t_5}\\ \overrightarrow{\mathbf{d_3}} &= 0\hat{t_1} + 1\hat{t_2} + 0\hat{t_3} + 1\hat{t_4} + 2\hat{t_5}\end{aligned}$$

(Example 1.4)

where the document vector of i^{th} document is represented as $\overrightarrow{\mathbf{d_i}}$. In the Example 1.4 above, the components along each axis indicate the frequency of the term in the document. However, these components don't necessarily have to indicate frequency. As mentioned before, these components can indicate any association between the term and the document, for example, the importance or simply presence of a term in that document. Such a model of document representation where each axis of the document vector space represents a term is popularly known as *Vector Space Model*. However, the term Vector Space Model is too general and can be interpreted to refer to modelling of documents in any vector space. But in Vector Space Model, the vector space is specifically defined such that the axes of the vector space represent the terms. In the rest of the chapter, other document vector spaces will be discussed where each axis does not represent a term. Therefore this term, Vector Space Model, is not specific enough to differentiate between the different document vector spaces. So for the purpose of differentiating between various vector spaces discussed in this chapter, I have and I will continue to refer to it as *Document-Term Vector Space Model* (*DT-VSM*) instead. Moreover, since the document vectors in Example 1.4 indicate only the terms and their association with the document, I'll refer to such vectors as *Document-Term Association Vector* in this thesis.

> *Document-Term Vector Space Model* is the model wherein a document is represented as a vector such that each axis of the vector represents a term and the corresponding component/weight indicates a numerical association of the term with the document. In this thesis, each vector in this vector space is referred to as *Document-Term Association Vector*.

(Definition 2)

Similarly, though the Document-Term Matrix is defined to contain the frequency of terms in documents, the association between a term and a document doesn't have to be limited to the frequency. Any *document-term association data* can be represented in a similar matrix where the cells indicate a numerical association of the term with the document.

> A *Document-Term Association Matrix* is a document by term matrix which gives the association of each term in every document of the dataset. In a Document-Term Association Matrix, each row represents a document, each column represents a term and each cell at the intersection indicates a numerical association of the term in the document.

(Definition 3)

When the association between a term and a document is binary, it indicates the presence (1) or the absence (0) of the term in the document. For example, the documents in Example 1.2 can be represented in a *Binary Document-Term Association Vector* as -

$$\begin{aligned}
\overrightarrow{\mathbf{d_1}} &= 0\hat{t_1} + 0\hat{t_2} + 1\hat{t_3} + 1\hat{t_4} + 1\hat{t_5} \\
\overrightarrow{\mathbf{d_2}} &= 1\hat{t_1} + 1\hat{t_2} + 0\hat{t_3} + 0\hat{t_4} + 1\hat{t_5} \\
\overrightarrow{\mathbf{d_3}} &= 0\hat{t_1} + 1\hat{t_2} + 0\hat{t_3} + 1\hat{t_4} + 1\hat{t_5}
\end{aligned}$$

(Example 1.5)

In *Binary Document-Term Association Vector*, the contribution of the terms is reduced to a mere presence or absence in the document. However, the frequency of a term in the document may better indicate the contribution of that term in the document. But in the vector representation based on the frequency of terms, every single occurrence in the document is attributed equal importance. So, if *cancer* was as frequent in a document as *the*, the axes representing both the terms will be weighted equally. Practically, *the* is likely to be more frequent. Therefore, according to frequency based Document-Term Association Vector, *the* weighs more in the document than cancer *cancer*. This ultimately leads to a representation of the document where each occurrence, i.e. each token, of both the terms contributes equally to the representation, while that may not be ideal. As we can see, some words, for example, *the*, *a*, *and*, are very common and do not help in distinguishing between two documents. Such words in a language, that are very common and contribute little to nothing to the distinctive aspects of documents are known as *stopwords*.

Therefore, it is often desirable that the association between the terms and documents be indicative of the relevance of that term in the document. To this end, the weight of a term in the document can be calculated using a weighting scheme which can, hopefully, indicate its relevance in the document. *Term frequency-inverse document frequency (TF-iDF)* is one such weighting scheme that assigns a weight to a term t in the document d based on the the term's frequency in the document as well as the term's prevalence the entire corpus. This scheme introduces a mechanism to attenuate the weight of terms that are too prevalent in the corpus to contribute a distinctive value to the representation of the document. In other words, on top of accounting for the frequency of a term in the document (term frequency), the score is also designed to account for the rarity of its usage across various documents. Therefore the score of term for a document is highest when that term rarely appears in any other document, but appears frequently in this document. The TF-iDF score is proportional to the *term frequency* in the document and attenuates the score by the means of *inverse document frequency*. The formula for calculating TF-iDF score for term t in document d is given below in Equation (2.1) -

$$\begin{aligned} \text{tf-idf}_{t,d} &= \text{tf}_{t,d} \times \text{idf}_t \\ &= \text{tf}_{t,d} \times \log\left(\frac{N}{\text{df}_t}\right) \end{aligned} \tag{2.1}$$

where, $\text{tf-idf}_{t,d}$ refers to the TF-iDF weight of term t for document d and the idf_t is the *inverse document frequency* of term t. Additionally, *term frequency* $\text{tf}_{t,d}$ refers to the frequency of term t in document d and df_t refers to the number of documents, out of total number of documents N, that the term t appears in. The TF-iDF weighting scheme can be used to calculate a score to indicate the numerical association of term with the document. The Document-Term Association Vector using TF-iDF weights can be referred to as a *TF-iDF Document-Term Association Vector*, or simply TF-iDF vectors.

In TF-iDF weighting scheme, the document frequencies of the terms are calculated on a dataset. However, for a new term t' that did not exist in the dataset on which the TF-iDF values were calculated, the document frequency of term t' will be zero, i.e. $df_{t'} = 0$. Therefore, this can lead to division by zero in the calculation of inverse document frequency term, i.e. $\text{idf}_{t'}$. Since the division by zero is undefined, the TF-iDF score can be slightly modified to take such a case into account. One way to avoid division by zero is to assume an additional document that contains all the terms exactly once. This way, for any term there is at least one document that contains the terms and $df_{t'} \neq 0$. As a result of this, the total number of documents is increased by 1, i.e. $N + 1$. This modified inverse document frequency is known as *smoothened inverse document frequency*. The Equation (2.1) with smoothened inverse document frequency is as given below -

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \log\left(\frac{N+1}{\text{df}_t + 1}\right) \tag{2.2}$$

To recap, in this section, I've discussed Document-Term Association Vector and different weighting schemes that can be used to create a document vector using the Document-Term Vector Space Model. However, there are a few things to consider when representing a document as a Document-Term Association Vector. In a Document-Term Association Vector, since each term is an axis, the dimensionality of the vector space can be of the order of vocabulary of the dataset. Any document represented in this high-dimensional vector space is likely to be sparse. Even for a dataset of news articles with nearly 2000 terms in each article and a vocabulary of 50,000 terms, any vector representing a document will have no component corresponding to nearly 48,000 axes. Such high sparsity in data can reduce the performance of the statistical AI methods. Therefore, often only a limited number of most frequent terms are used to represent the documents and the rest are discarded. This leads to an increase in performance, however, by trading off a significant portion of the vocabulary.

More importantly, most of the terms are words and words have a meaning associated with them in the language. However, in Document-Term Vector Space Model, a document only indicates the association of each term with the document without accounting for the similarity among the terms. In other words, in Document-Term Vector Space Model, since each word is represented as an axis in this vector space, each term is a separate entity and any relationship between the terms goes completely unrepresented. For example, let's consider the documents below -

			likes	*loves*	*cooks*	*rain*	*snow*	*dog*
d_1	:	she likes the rain	1	0	0	1	0	0
d_2	:	she loves the snow	0	1	0	0	1	0
d_3	:	that dog cooks	0	0	1	0	0	1

(Example 2.1)

The table above shows a Binary Document-Term Association Vector for three documents. In the example above, the vectors representing all three documents are equally distant in the Document-Term vector space. That is, this representation of the document completely bypasses any similarity that exists between the terms *likes* and *loves*, or between the terms *rain* and *snow*. According to this representation, there is no reason why *she likes the rain* is more similar to *she loves the snow* than *that dog cooks*. In this representation, a document is completely stripped off of the meaning of the words, and only the record of occurrence of individual term remains. But the meaning of the words plays a significant role in making the document what it is. Since the goal of the document representation is to map the similarity between documents to a vector space, the representation must transcend mere term occurrence and encode within it some sense of meaning.

2.2 Document-Topic Vector Space Model

A document is constituted of one or more themes or topics. Based on this intuition a certain set of techniques attempt to use the *document-term association data*, like a *Document-Term Matrix*, to derive a representation of the documents where each document is hopefully represented as a distribution over topics. These techniques transform the *term-document associations* that we discussed in the last section, to a new dimensional space where each axis is observed to be a representative of an *artificial concept* akin to a theme or topic. Therefore, these techniques have come to be popularly known as *topic modeling techniques*. For example, consider the document vector $\overrightarrow{\mathbf{d}}$ given below -

$$\overrightarrow{\mathbf{d}} = 0.22\hat{q_1} + 0.05\hat{q_2} + 0.57\hat{q_3} + 0.13\hat{q_4} + 0.03\hat{q_5}$$
$$\overrightarrow{\mathbf{q_1}} = 0.12\hat{t_1} + 0.05\hat{t_2} + 0.07\hat{t_3} + 0.01\hat{t_4} + \ldots + 0.03\hat{t_m}$$

(Example 3.1)

where, an arbitrary document $\overrightarrow{\mathbf{d}}$ is represented in terms of 5 axes, where each axis $\hat{q}$ corresponds to a *topic*. The *topics* $\overrightarrow{\mathbf{q}}$, in turn, form a distribution over the $t_1, t_2, t_3, ..., t_m$ terms. In this vector space, each document vector gives an association of a document with each of the artificial concepts or topics. Therefore, in this thesis, such models will be collectively referred to as *Document-Topic Vector Space Model* and a vector in this space will be referred to as the *Document-Topic Association Vector*. In this section, I will discuss two such popular topic modeling techniques that can be used to represent the documents as a distribution over topics - Latent Semantic Analysis and Latent Dirichlet Allocation.

2.2.1 Latent Semantic Analysis (LSA)

> "Many mathematical objects can be understood better by breaking them into constituent parts, or finding some properties of them that are universal, not caused by the way we choose to represent them.…"
>
> "…Much as we can discover something about the true nature of an integer by decomposing it into prime factors, we can also decompose matrices in ways that show us information about their functional properties that is not obvious from the representation of the matrix as an array of elements."
>
> *(Goodfellow et al. (2016))*

The passages quoted above beautifully explain the value of decomposing mathematical objects, and this forms the basis of the process of Latent Semantic Analysis.

In the previous section, we've discussed the representation of the association between the term and a document in the form of a matrix, i.e. a *Document-Term Association Matrix*. The *Document-Term Association Matrix* defined in the previous section refers to a matrix where each cell indicates a numerical association of a document (row) with the term (column). This matrix records a numerical association between the documents and the terms. However, this matrix records no association among the terms. Each term is treated as a completely different feature of the text than the rest. However, as we see in Example 2.1, often different words in the language aren't wholly unrelated.

Latent Semantic Analysis hopes to encode the relatedness between the terms in a vector space. It uses the association data between term and document to construct an artificial space where the closely related terms are placed near each other in the vector space. In the artificial vector space created by Latent Semantic Analysis, each axis/feature is referred to as an *artificial concept* and represents a semantic structure akin to a topic. Each vector in this vector space represents a distribution over the topics. In Latent Semantic Analysis, both the documents and the terms are represented in the same vector space, i.e. in terms of the topics. Most importantly, through the Latent Semantic Analysis, both documents and terms, are placed in this artificial space such that the closely related terms and documents are closer in this space as well. For example, let's extend Example 3.1 to show the representation of documents and terms in the same space. Example 3.1 shows an arbitrary document d represented in terms of artificial concepts $q_1, q_2, q_3, ..., q_5$. Similarly, in Latent Semantic Analysis an arbitrary term t_1 is represented in the same space, i.e. in terms of the same artificial concepts as the document d -

$$\overrightarrow{\mathbf{d}} = 0.22\hat{q_1} + 0.05\hat{q_2} + 0.57\hat{q_3} + 0.13\hat{q_4} + 0.03\hat{q_5}$$
$$\overrightarrow{\mathbf{t_1}} = 0.12\hat{q_1} + 0.33\hat{q_2} + 0.37\hat{q_3} + 0.08\hat{q_4} + 0.10\hat{q_5}$$

(Example 3.2)

where, $\overrightarrow{\mathbf{t_1}}$ refers to the vector of a term t_1 as a distribution of artificial concepts/topics $q_1, q_2, q_3, ..., q_5$. The weight corresponding to an artificial concept in the vector indicates the strength of the association of the term or document with respect to that artificial concept (Deerwester et al., 1990). At this point we can define Latent Semantic Analysis in terms of its process. -

> *Latent Semantic Analysis* is a technique that uses truncated Singular-Value Decomposition (SVD) to decompose the *term by document association matrix* and represent the terms and documents in an artificial semantic space such that the closely related terms and documents are closer in this space as well (Deerwester et al., 1990).
>
> *(Definition 4)*

As to how Latent Semantic Analysis constructs the artificial space, let's take a closer look at the said process. Latent Semantic Analysis uses a term by document association matrix, i.e. *Term-Document association matrix*, to construct this space. The term by document association matrix is the association data between the terms and documents presented in the form of a matrix where the rows indicate the term and columns indicate the documents. It is simply the transpose of *Document-Term association matrix* and, for LSA as well, this matrix could indicate incidence, frequency, or importance. Let's call this matrix containing the term by document association data $\mathbf{M}$, regardless of the numerical association contained in the matrix. For further clarity, let's suppose we have N documents and a vocabulary of V terms. Then, an entry $\mathbf{M}_{ij}$ in the matrix $\mathbf{M}$ corresponds to a score representing an association or weight of term i in document j. For example, a binary term by document association matrix for the documents in Example 1.2 will be -

Document by term
binary association matrix ($\mathbf{M}^{\mathrm{T}}$)

$$\begin{array}{c|ccccc} & t_1 & t_2 & t_3 & t_4 & t_5 \\ d_1 & 0 & 0 & 1 & 1 & 1 \\ d_2 & 1 & 1 & 0 & 0 & 1 \\ d_3 & 0 & 1 & 0 & 1 & 1 \end{array}$$

Term by document
binary association matrix ($\mathbf{M}$)

$$\begin{array}{c|ccc} & d_1 & d_2 & d_3 \\ t_1 & 0 & 1 & 0 \\ t_2 & 0 & 1 & 1 \\ t_3 & 1 & 0 & 0 \\ t_4 & 1 & 0 & 1 \\ t_5 & 1 & 1 & 1 \end{array}$$

In Latent Semantic Analysis, using Singular Value Decomposition (SVD), the term-document association matrix $\mathbf{M}$ is decomposed into three matrices, i.e. $\mathbf{T}$, Σ and $\mathbf{D}$, as shown below -

$$\mathbf{M} = \mathbf{T}\,\Sigma\,\mathbf{D}^{\mathrm{T}} \tag{2.3}$$

$$\begin{bmatrix} \\ \\ \quad \mathbf{M} \quad \\ \\ \\ \end{bmatrix}_{V\times N} = \begin{bmatrix} \\ \\ \quad \mathbf{T} \quad \\ \\ \\ \end{bmatrix}_{V\times V} \begin{bmatrix} \\ \\ \quad \Sigma \quad \\ \\ \\ \end{bmatrix}_{V\times N} \begin{bmatrix} \\ \quad \mathbf{D}^{\mathrm{T}} \quad \\ \\ \end{bmatrix}_{N\times N} \tag{2.4}$$

These three matrices contain a breakdown of the original associations into new artificial concepts. In Equations (2.3) and (2.4), the left matrix $\mathbf{T}$ is a $V \times V$ matrix where each row is a vector representation of a term as a distribution over the artificial concepts or topics. Similarly, the right matrix $\mathbf{D}$ is a $N \times N$ matrix where each row is a vector representation of a document as a distribution over the artificial concepts or topics. The Σ matrix is a diagonal matrix that contains values, called the *singular values*, that indicate the influence of the artificial concept in the latent semantic space. These values are ordered in the decreasing order of magnitude. That said, for some artificial concepts, the singular values are really small. Removing these artificial concepts reduces the dimensionality of this artificial vector space and optimally approximates the original association data (Deerwester et al., 1990; Manning et al., 2008d). Therefore, in the next step, the artificial concepts that have the least influence are ignored. That is, the largest k singular values on the diagonal of Σ are retained and the rest of the matrix is truncated. Here, k is the desired number of topics, i.e. the dimensionality of the artificial vector space. The truncation results in a matrix Σ_k which is a square diagonal matrix of $k \times k$ dimension. For example, let's consider an arbitrary Σ matrix. Then the Σ can be truncated for $k = 3$ as shown below -

$$\Sigma = \begin{bmatrix} 0.20 & & & & \\ & 0.11 & & & \\ & & 0.07 & & \\ & & & 0.03 & \\ & & & & 0.01 \\ & & & & \\ & & & & \end{bmatrix}_{7\times 5} \qquad \Sigma_3 = \begin{bmatrix} 0.20 & & \\ & 0.11 & \\ & & 0.07 \end{bmatrix}_{3\times 3}$$

In the matrices above, the matrix on the left shows arbitrary singular values where the empty cells are all zero, and the matrix on the right shows its truncated version with only the largest 3 singular values retained.

Just like the Σ above, the $\mathbf{T}$ and $\mathbf{D}$ are also truncated such that only the first k columns are retained. Let the truncated matrices be represented by Σ_k, $\mathbf{T}_k$ and $\mathbf{D}_k$, respectively. Now, for a new pseudo document, these truncated matrices are used to find a representation of the document in terms of the k artificial concepts. In order to do so, a document is first represented in terms of a Document-Term Association Vector with the same numerical association function as the one used for the Term-Document Association Matrix $\mathbf{M}$ in the beginning. Then this new pseudo-document d can then be transformed from its Document-Term Association Vector form to the LSA Document-Topic Association Vector representation using Equation (2.5) -

$$\mathbf{d_k} = \Sigma_k^{-1} \, \mathbf{T}_k{}^{\mathrm{T}} \, \mathbf{d} \tag{2.5}$$

where, $\mathbf{d}$ refers to the Document-Term Association Vector of document d and $\mathbf{d_k}$ refers to the representation of document as a distribution over k artificial concepts. This Document-Topic Association Vector representation $\mathbf{d_k}$, which is obtained from the term-document vector $\mathbf{d}$, is considered to be the LSA Document-Topic Association Vector vector of document d.

In the end, it is important to note here that unlike the Document-Term Vector Space Model, an axis in the Document-Topic Vector Space Model does not represent a term. Rather, each axis/feature is now replaced by new features as the descriptors of documents. These new features, in turn, can be described in terms of the terms or the documents they are associated with (Deerwester et al., 1990). Latent Semantic Analysis leverages the document co-occurrence between the terms to establish relatedness between the terms in terms of the artificial concepts. Similarly, it uses the term co-occurrence between documents to establish the relatedness between the documents in terms of the artificial concepts. Latent Semantic Analysis does so by using the Singular-Value Decomposition to arrange an artificial space that reflects the major associative patterns in the term and document association data. However, this is only one of the ways to represent terms and documents in terms of a distribution of artificial concepts.

2.2.2 Latent Dirichlet Allocation (LDA)

Just like Latent Semantic Analysis, in Latent Dirichlet Allocation, the documents are also represented as a distribution over topics. The topics, in turn, are defined as a distribution over terms in the vocabulary (Example 3.1). However, unlike LSA, LDA is a probabilistic model.

> *Latent Dirichlet Allocation* is generative probabilistic modelling technique which approximates the *posterior* or conditional probability of the hidden topics in the documents given the terms observed in the documents.
>
> *(Definition 5)*

In LDA, it is assumed that the documents arose from an imaginary random process. Before we discuss the process, let's discuss the assumptions. Let us consider a dataset with N documents and a total vocabulary of size V. In the generative process, we assume a number of topics that each document is distributed on. This is the desired number of topics we hope to represent the document in. Let's say that the desired number of topics is K. The documents are assumed to be a discrete probability distribution[1] over K topics. Similarly, the topics are assumed to be a discrete probability distribution over V terms. Therefore, there are two distributions assumed in this process, i.e. θ_d and β_k -

1. Distribution of topics in documents, $\theta_d \sim \text{Dirichlet}(\alpha)$, and,
2. Distribution of terms in topics, $\beta_k \sim \text{Dirichlet}(\eta)$,

where, θ_d refers to the distribution of K topics in document d and β_k refers to the distribution of V terms in topic k. Additionally, α and η are the parameters which define the Dirichlet priors from which the two distributions, θ_d and β_k, are drawn, respectively (Hoffman et al., 2010).

Now, to understand the overall assumed generative process, let's take the example of a document d with a distribution of topics given by θ_d. Then, to generate n^{th} term for document d -

1. Draw a topic index $x \in \{1...K\}$ from the topic weights $x \sim \theta_d$.
2. Draw the n^{th} observed term of document d from the selected topic $t_{d,n} \sim \beta_x$.

where $t_{d,n}$ is the observed term in the document and β_x is the distribution of terms in the topic x.

This generative process defines a *joint probability distribution* over both the hidden topics structures and the observed terms in the document. This joint probability distribution is then used to compute the conditional probability distribution of topics given the terms in the documents. However, the posterior or the conditional probability can not be computed directly and is usually only approximated (David M Blei et al., 2003; David M. Blei, 2012; Hoffman et al., 2010). The actual approximation is beyond the scope of the discussion of the thesis. However, it is important to note here that Latent Dirichlet Allocation of documents results in the representation of documents as a probability distribution over topics. In the vector form of this distribution, each axis is an artificial concept, like LSA, and the corresponding component indicates a weight, an association, of the artificial component with the document. Only that, in the case of LDA, the association is a probability.

Let's summarize the Document-Topic Association Vector representation of document. In both the topic modeling techniques discussed so far, the axis for document vectors is an artificial concept mathematically inferred from the distribution of terms in the documents. Each artificial concept itself is further defined as a vector where each axis is a term and the corresponding component is a weight of the term in the definition of the artificial component. An example of this is shown in Example 3.1, which is repeated below -

$$\overrightarrow{\mathbf{d}} = 0.22\hat{q_1} + 0.05\hat{q_2} + 0.57\hat{q_3} + 0.13\hat{q_4} + 0.03\hat{q_5}$$
$$\overrightarrow{\mathbf{q_1}} = 0.12\hat{t_1} + 0.05\hat{t_2} + 0.07\hat{t_3} + 0.01\hat{t_4} + \ldots + 0.03\hat{t_m}$$

[1]The probability of each value in the distribution is in the range $[0, 1]$ and the sum of all probabilities for a distribution is 1.

where, $\overrightarrow{\mathbf{d}}$ is the Document-Topic Association Vector of document d in terms of artificial concepts $q_1, q_2, q_3, ..., q_5$ and $\overrightarrow{\mathbf{q_1}}$ is representation of the artificial concept q_1 as a distribution over all the terms in the vocabulary $t_1, t_2, t_3, ..., t_m$.

However, it is important to note that for both of the methods, the mathematical model is based on the occurrence of terms in documents. Therefore, in these methods, two terms are only related based on the document co-occurrence. However, the words in a language have a meaning, a similarity, a relatedness with each other. But with respect to the relationship between the terms, the topic modeling techniques discussed in this section do not go beyond the document co-occurrence of terms. Any relationship established between the terms is in the context of the entire document (Socher, 2015). Let's see why this matters.

Let's consider a document. If all the words in this document were switched out by their near-synonyms, Document-Term Association Vector of the document would reflect a completely different document. However, if these synonyms were previously observed in the dataset during the topic modeling, the Document-Topic Association Vector of the document may yet represent them as a part of the same topic. However, if all the words in the document were switched out by their near-synonyms that have never been observed in the dataset, then the Document-Topic Association Vector will not be able to associate the near-synonyms based on the document co-occurrence. But it still does not change the fact that the words are related and very close in meaning. But any information about the universal meaning for words is not taken into account while representing documents as either Document-Term Association Vector or Document-Topic Association Vector. Moreover, according to the *distributional hypothesis*, similar words occur in similar contexts (Jurafsky et al., 2021). However, the context for the similarity between terms in Document-Topic Association Vector is the entire document, which may be too broad for effectively associating two words with respect to their similarity in meaning (Socher, 2015).

A set of techniques use a smaller neighboring context to compute representations that encode some aspect of the meaning of a word. In such techniques, a word is represented in a vector space where the representation of the word is computed based on its neighboring words throughout the documents. The representation so learnt places the words in the vector space such that the vectors that are used in similar contexts are closer in the vector space. These representations can be computed over a large amount of data to reflect their general similarity with respect to other words in the language. Once computed, these representations of words can stand on their own for comparing different words in the language and establishing a similarity between them. Since, a word is a basic unit in the document which significantly contributes to the overall meaning imparted by the document, using such word representations can allow us to encode some extent of meaning imparted by the document in its vector representation. In the next section, I'll discuss such representation of words, and subsequently, representation of documents based on such representation of words.

2.3 Document representations using Word Vectors

Before we discuss the representation of documents using word vectors, let us briefly discuss the vector representations of words. The vector representation of words refers to the representation of word-level units of text documents. Any representation of a word in the form of a vector can be called a *word vector*. However, there are different types of vector spaces for the representation of a word as a vector. Therefore, a *word vector* can refer to different representations of words depending on the vector space assumed. In the discussion that follows I'll briefly discuss a few of these vector spaces for the representation of words.

2.3.1 Word Vectors

Words as Document Association Vectors Just like a document can be represented in terms of the words, a word can be represented in terms of the document it appears in. In the case of such a word vector, each axis of the vector represents a document. As explained in Section 2.1, the association between a word (term) and document can be represented as a matrix called Document-Term Matrix (Definition 3, page 9). An example of a Document-Term Matrix given in (Example 1.3) is-

$$\begin{array}{c|ccccc} & t_1 & t_2 & t_3 & t_4 & t_5 \\ \hline d_1 & 0 & 0 & 2 & 1 & 2 \\ d_2 & 1 & 1 & 0 & 0 & 1 \\ d_3 & 0 & 1 & 0 & 1 & 2 \end{array}$$

Each row of this matrix is a document vector that represents the document in terms of the vocabulary of the dataset. At the same time, each column of this matrix is a word vector that represents the word in terms of the documents it appears in. In this way, a word can be represented in terms of the documents it appears in.

Words as Term Association Vectors Quite similar to Document-Term Association Vector, a word can also be represented in terms of other words. For representing words in terms of other words, we can construct a $V \times V$ where V refers to the total number of words in the dataset. Instead of the document and word associations in Document-Term Matrix, this matrix records the association between two words of the vocabulary. For simplicity, let's indicate the frequency of their document co-occurrence in the matrix. In other words, for association score between two words, let's record the number of documents in which both of the words occur. Therefore, each cell in this $V \times V$ matrix represents the number of times the words represented by the intersecting row and column occurred in the same document. For example, let's assume we have a vocabulary of the five words as given in the table below -

	win	match	thriller	horror	movie
win	23	11	3	3	5
match	11	31	0	0	1
thriller	3	0	19	15	19
horror	3	0	15	25	23
movie	5	1	19	23	38

(Example 5)

In the example above, both the row and columns indicate a word in the vocabulary and the cell at the intersection indicates the frequency of co-occurrence of the words in documents. So, for example, according to the above table, there are 15 documents in which the words *thriller* and *horror* appeared together. However, we could also limit these statistics to a few neighboring words. For example, let's say that we only look at the 5 words before and 5 words after the word *thriller*. Then, for *thriller*, we only count occurrences of the word *horror* if it appears in these "neighboring" 10 words. These 5 words before and 5 words after the *target word*, i.e. *thriller*, form a *context window*. In this case, we have limited the context we observe around each of our *target words* to a window of 5. This becomes significant in developing techniques that encode the contextual co-occurrence, and some aspect of word meaning in the vector representation. Those techniques will be discussed next.

Semantic Word Vectors

"You shall know a word by the company it keeps."

(Firth (1957))

The next class of word vectors attempts to encode the contextual information, i.e. company a word keeps, into the vector representation of a word. However, unlike the word vectors where the observed context of a word is the entire document, these techniques limit the observed context for each word to a few neighboring words. This allows for capturing statistics that draw out the words that often keep the same company of words, and are thereby similar. Ultimately, building upon this intuition, these techniques attempt to place each word in an artificial vector space such that the words that are used in similar contexts, and are therefore similar, are placed near one another. The word vectors so obtained incorporate the aspects of word meaning that can be inferred from the context of words. Since these vectors are designed to embed some aspects of the meaning of words in a language, let's call them *Semantic Word Vectors*[2].

The word vectors that incorporate some aspects of the meaning of the words, usually the aspects that can be inferred from their contexts, are referred to as *Semantic Word Vectors* in this thesis. The entire structure that collectively associates words with their respective semantic word vectors is known as *Word Embedding*.

(Definition 6)

[2]"Semantic" here refers to the meaning of a word only with respect to the fact that context embeds within it some aspect of the meaning of a word.

For a class of semantic word vectors, while the context is taken into account while computing (learning) the representation of words from the data, ultimately the representation computed (learnt) for each word is fixed. These semantic word vectors that have a fixed vector representation for each word are considered *static*, and their embedding is called *static word embedding* (Jurafsky et al., 2021). However, a word in a language can have multiple meanings. For example, consider the sentences below -

1. I finally found the first edition of that book.
2. I am planning to book two seats for the show.
3. The police will finally book him for murder.
4. I like doing things by the book.

(Example 6)

In the above example, the word *book* has been used in different senses. Though the learning process will take all the instances of *book* into account, in the case of static semantic word vectors, the representation ultimately learnt for *book* will be fixed. That is, when we look for a representation of *book* in static semantic word vectors that have been computed from data, we will get 1 representation, which will be the same for all the cases above. However, in each case above, though the word is the same, its referent is not. Here's where another class of semantic word vectors takes the context into account while representing the word. That is, there may be a different vector representation of the word *book* in each of the cases above, depending on its usage, its context. This class of word embedding, e.g. ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), that generate *contextualized* semantic word representation based on the context the word is being used in, are called *contextualized word embeddings* (Jurafsky et al., 2021). That said, the scope of discussion and analysis in this thesis is limited to only static word embeddings, specifically - Word2vec and GloVe. In the following sections, I'll discuss both of these word embedding in just as much detail as necessary for the upcoming discussion in the following chapters. Let's first discuss Word2vec.

Word2vec

Mikolov et al. (2013) introduced two model architectures for estimating semantic vector representations of words from large datasets - continuous bag-of-words (CBOW) and skip-gram models. These models are known as *Word2vec* model architectures. In both architectures, there is a *target word* and a set of *context words*. For example, in the sentence -

$\boxed{\text{As}}$ $\boxed{\text{I}}$ $\boxed{\text{walked}}$ $\boxed{\text{past}}$ $\boxed{\text{the}}$ $\boxed{\text{lake}}$ $\boxed{\text{that}}$ $\boxed{\text{reflected}}$ $\boxed{\text{the}}$ $\boxed{\text{burning}}$ $\boxed{\text{autumn}}$ and the radiating gray sky with silver edges, the hours went by and my heart stood still.

(Example 7)

lake is the *target word* (shown by the colored box) with a context window of 5 words (shown by black border boxes). That means that the 5 words before and after the target word are considered *context words*. The *Continuous Bag-of-words (CBOW)* architecture is designed such that the model is trained while predicting the target word based on the given context words. That is the model takes the context words as input and predicts the target word as the output. For example, predicting the target word *lake* based on the 10 context words given in Example 7 forms one *training sample* for Continuous Bag-of-words model architecture. On the other hand, in the *Skip-gram* architecture, the model is trained while predicting the context words based on the target word. The task in Skip-gram architecture is formulated as a classification task based on whether, for each pair of words, they appear within the context window of each other. Though the exact mathematical formulation of Word2vec is beyond the scope of the thesis, an understanding of the process of word vector computation is essential in getting a sense of why it works the way it does. Moreover, a basic understanding of Word2vec is essential for understanding a neural-network-based document representation technique which was significantly inspired from Bengio et al. (2003) and Mikolov et al. (2013) which will be discussed later in Section 2.4. Most importantly, it develops the intuition which is necessary to analyze the methodology discussed in Chapter 3 with respect to the extents and limits of the semantic word vectors so computed. Therefore, we'll discuss the word vectors in just enough detail to develop a basic understanding of the process. Please note that the explanation below assumes a working understanding of the key components of Artificial Neural Networks (ANN) - the objective function (or the loss function), weight update mechanism (backpropagation), and the network architecture (Richards et al., 2019).

Figure 2.2[3] shows the model architectures of both CBOW and Skip-gram where both model architectures contain 3 layers - input, projection, and output. Before we discuss the process, let's discuss a few important variables in Figure 2.2. In the figure, V is the size of our vocabulary, N is the size of the projection layer, and also the desired size of a word vector. The *input weight matrix*, i.e. the weight matrix between the input layer and the projection layer is represented by $\mathbf{W} \in \mathbb{R}^{V \times N}$. Similarly, $\mathbf{W}' \in \mathbb{R}^{N \times V}$ represents the output weight matrix, i.e. the weight matrix between the projection layer and the output layer[4]. Input weight matrix $\mathbf{W}$ has V rows such that each row is an N-dimensional vector corresponding to a word in the vocabulary. Similarly, the output weight matrix $\mathbf{W}'$ has V columns such that each column of $\mathbf{W}'$ is an N-dimensional vector corresponding to a word in the vocabulary.

In the CBOW training process, the target word is predicted based on the context word. Therefore, for CBOW, the input word vectors given in $\mathbf{W}$ are considered *context word vectors*, or *context embedding*, since it contains semantic word vectors for context words. Similarly, the output word vectors given in $\mathbf{W}'$ in CBOW are considered the *target word vectors*, or *target embedding*, as it contains semantic word vectors for the target word. In Skip-gram, on the other hand, the input word vectors given in $\mathbf{W}$ are considered *target embedding*, and the output word vectors given in $\mathbf{W}'$ are considered the *context word vectors*.

The matrices $\mathbf{W}$ and $\mathbf{W}'$ matrices may be randomly initialized but are updated through the training process. These matrices are not defined separately for each training sample but are rather shared across all the training samples (Figure 2.2). Therefore, each training sample during the training uses and updates both of these matrices. While these matrices $\mathbf{W}$ and $\mathbf{W}'$ are updated in the training process, at the end of the training they reflect the semantic representation of words

[3]The figure and the notation has been borrowed from Rong (2014) and have been slightly modified for our discussion
[4]Note: $\mathbf{W}'$ does not represent transpose of $\mathbf{W}$, but instead represents a different matrix altogether.

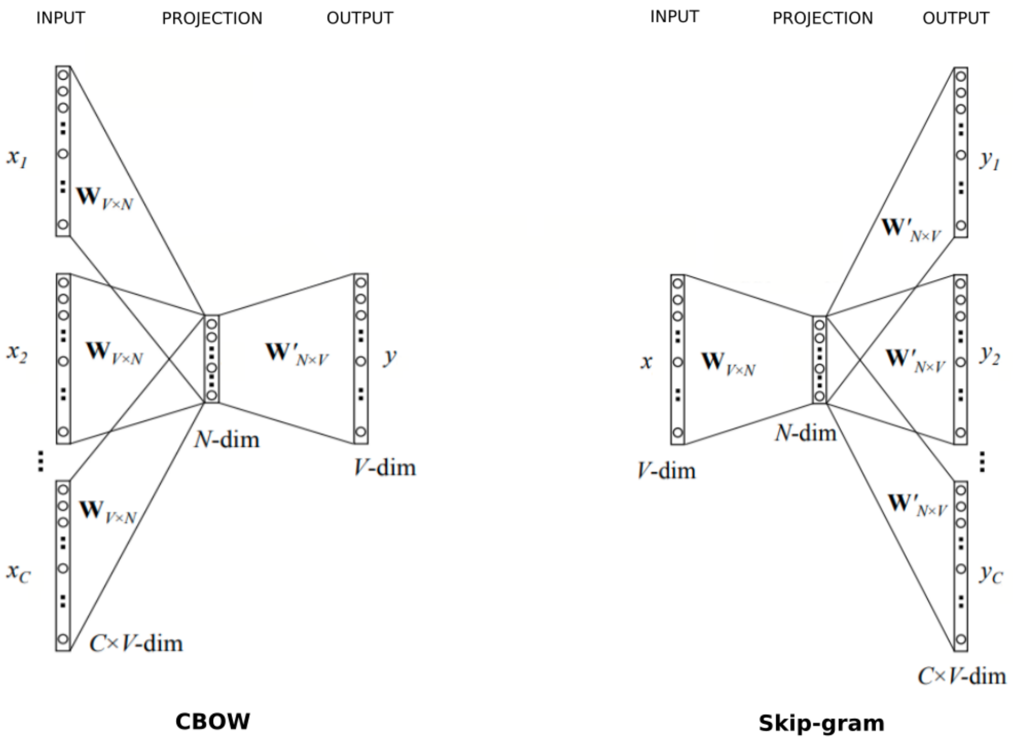


Figure 2.2: Word2vec model architectures proposed by Mikolov et al. (2013) as explained in Rong (2014) (slightly modified). In CBOW architecture *(left)*, the current word is predicted given the context words, whereas in the Skip-gram architecture *(right)* the neighboring context words are predicted given the current word.

learnt in the Word2vec process. Ultimately, since two word embeddings are trained in the process of Word2vec, for each word in the vocabulary, we learn two sets of semantic word vectors (Socher, 2015). However, the final semantic word vector for a word could be the sum of both, or alternatively, we can also use just the target word embedding, i.e. $\mathbf{W}'$, to obtain final semantic word vector representations (Jurafsky et al., 2021). Now let's take a look at the process to understand how the weight matrices are updated.

For clarity of the training process of Word2vec, let's consider one pass of the CBOW model at the moment, i.e. let's predict the target word given the context words. To prepare the input to the model, the words are first represented in a *one hot vector* form. A *one hot vector* of a word is a vector of the size of vocabulary, V, such that each index in the vector corresponds to a word in V, just like Document-Term Vector Space Model. However, since we are representing words with this representation, only the index corresponding to the word will be 1, rest will be 0, making it a 1-hot representation of words. For example, let's consider a vocabulary of 5 terms from Example 1.1 - *won*, *lost*, *ran*, *100m* and *race*. So to represent the word *lost* as a 1-hot vector based on this vocabulary, for example, only the index corresponding to the word *lost* will be 1 and the rest will be 0 as shown below -

won	*lost*	*ran*	*100m*	*race*
0	1	0	0	0

Similarly, the input to Word2vec model, i.e. context words, are words represented in the form of a 1-hot vector. So for a total of C context words, let x_i be the one-hot encoded vector of each of the context words, where $i \leq C$. Therefore, the input is a matrix of size $C \times V$. Let's call this input matrix $\mathbf{I} \in \mathbb{R}^{C \times V}$. We should note here that the matrix multiplication of input matrix $\mathbf{I}$ and the input weight matrix $\mathbf{W}$ gives the semantic word vectors embedded in the context embedding $\mathbf{W}$ corresponding to each of the context words. In the forward pass from the input layer to the projection layer, these context word vectors are averaged to obtain an N sized vector corresponding to the projection layer. Since the model is designed to predict the target word, the projection layer values are then passed forward through the output weight matrix $\mathbf{W'}$ to give a prediction score for each word in the vocabulary in the form of a V dimensional vector y. These scores are then converted into probabilities using a softmax[5] function which is expected to reflect the probability of each word being the target word given the context words. The predicted probability for each word is then compared with the true target hot vector and the cross-entropy loss is backpropagated using gradient descent to update the weight matrices $\mathbf{W}$ and $\mathbf{W'}$. This is how the weights are adjusted to best match the predictions and therefore ultimately can be used to represent the words themselves.

On the other hand, in Skip-gram architecture, for a given word, represented by its one-hot encoding x, the context words $\mathrm{y}_i, i \leq C$ are predicted. Analogous to the logic applied in CBOW architecture, in Skip-gram, $\mathbf{W}$ and $\mathbf{W'}$ are updated through the training process, where the former is considered the target embedding and latter the context embedding. In Skip-gram, given a pair of a target word and a context word the model predicts whether the context word appears within the context of the target word or not. But, if we only add training samples from the context window of each target word we will only have samples of the positive class. Therefore, to have training samples of the negative class, additional samples are added to the training set. To obtain samples with negative class a few noise words from the vocabulary, which are not a part of the context of the target word, are paired with the target word as a negative sample. This is commonly called *Skip-gram with Negative Sampling (SGNS)*. Overall, in both the models, the semantic word vectors are computed in a model designed to estimate the word vectors given their neighboring words or vice versa.

GloVe - Global Vectors

Another popular algorithm for computing word vectors is known as *GloVe* (Pennington et al., 2014). In Word2vec, any context word within the context window of the target word contributes to the computation of its representation. However, in GloVe the target and context word statistics are collected globally from the entire dataset, hence the name *Global Vectors* (GloVe). Therefore, unlike

[5]converts the distribution of scores into a discrete probability distribution, i.e. each probability value is in the range [0,1] and the sum of the probability values is 1.

Word2vec which computes the values on local context windows, GloVe computes word vectors on the basis of the global co-occurrence of words. To further clarify, I'll borrow the example of *ice* and *steam* from Pennington et al. (2014), albeit with different statistics for the sake of the example. Let's consider the following global co-occurrence of words -

		Context Words			
		solid	gas	water	fashion
Target words	ice	64	4	180	2
	steam	5	48	145	2

(Example 8.1)

The table above contains the co-occurrence of words in an arbitrary example. The context window is irrelevant for this example, but still, let's say that we select a context window of 5 words. Let's assume that only 4 words, i.e. *solid*, *gas*, *water* and *fashion*, ever appear within the context of given two words - *ice* and *steam*. Then, the cell at the intersection of *solid* (column) and *ice* (row) tells us that in the entire dataset, hence global, there were exactly 64 instances where *solid* was within the context window of *ice*. That is, there were 64 instances where *solid* was one of the 5 words before or after any occurrence of *ice*. These are the global statistics of contextual co-occurrence of words that form the basis of GloVe.

The mathematical foundation of GloVe is based on an important observation related to the ratio of probabilities that can be computed from the global contextual co-occurrence of words. To understand the observation, let's consider the probabilities calculated from the global co-occurrence statistics given in Example 8.1 -

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	0.256	0.016	0.720	0.008
$P(k\|steam)$	0.025	0.240	0.725	0.010
$P(k\|ice)/P(k\|steam)$	10.240	0.067	0.993	0.800

(Example 8.2)

In the table above, the $P(j|i)$ is calculated based on the frequency of the word j within the context of i. Let $\mathbf{X}$ be the matrix that represents the statistics given in Example 8.1 such that a cell $\mathbf{X}_{ij}$ represents the frequency of word j in the context of word i. Now given this, the $P(j|i) = \mathbf{X}_{ij}/\mathbf{X}_i$ where $\mathbf{X}_i$ refers to total number of times any word appeared in the context of word i. So, for *ice*, $\mathbf{X}_{ice} = 250$ and for *steam*, $\mathbf{X}_{steam} = 200$. As we can observe from the ratio of $P(k|ice)$ and $P(k|steam)$ given in the last row of the table, the words that relate exclusively to either *ice* or *steam*, but not both of them, have either very large value or very small value for ratio. On the other hand, the words that are either related to both *ice* or *steam*, or neither of them, have ratios close to 1. The words that relate to both or neither, i.e. *water* and *fashion*, do not contribute distinctive information with respect to the target words *ice* and *steam*. On the other hand, the words with very

Word embedding	**No. of words**	**Dimension**	**Training Dataset**
GloVe 100d	0.4 Million	100	Wikipedia 2014 and Gigaword 5 datasets (∼6 Billion tokens)
GloVe 300d	2.2 Million	300	Common Crawl (∼840 Billion tokens)
Word2vec Pre-trained	3 Million	300	Google News (∼100 billion tokens)

Table 2.1: A few pre-Trained Word Embeddings that are used in the experiments in this thesis

large or very small ratios, i.e. solid and gas, tell us not just that these words add distinction to the target words but also among themselves. Thus the fundamental intuition behind GloVe is that in comparison to raw probabilities, the ratio of co-occurrence probabilities can better encode some form of the meaning of words. Sparing the mathematical details, it should suffice for our purpose to state that the training objective of GloVe associates this ratio of co-occurrence probabilities with vector differences in the word vector space. Ultimately, in this way, based on the ratio of co-occurring probabilities, GloVe attempts to encode the word in a semantic vector space.

Though both of these algorithms, Word2vec and GloVe, can be used to compute word vectors from scratch from a dataset, some popular word embeddings are also available that have been trained over a large amount of data and are publicly available. These word embeddings are referred to as the *pre-trained word embeddings*. The word vectors computed over a specific dataset have the benefit of representing contextual information that is specifically pronounced in the corpus. However, having been computed over a large dataset, the semantic word vectors in pre-trained word embeddings often have the benefit of capturing more generalized contextual information corresponding to the words. The semantic word vectors can be obtained from a pre-trained word embedding, or they can be computed from scratch on a given dataset. Overall, the decision of whether to use pre-trained word embeddings or not largely depends on the task. In the experiments discussed in the thesis, I have used a selected few of the publicly available pre-trained word embeddings which were selected based on their vocabulary size, the training dataset, and the method used to compute the word vectors. Table 2.1 shows the size and training dataset details of a few of the pre-trained word embeddings trained using either GloVe[6] or Word2vec[7] model.

In this section, we started the discussion with the representation of words. However, instead of words, the focal point of this thesis is the representation of documents. The word representations discussed in this section provide the necessary background for the discussion of document representation methods in the rest of the thesis. Starting next section, we'll discuss the representation of documents in the form of a vector using the semantic word vectors that we've discussed in this section.

[6]Downloadable from `https://nlp.stanford.edu/projects/glove/` - GloVe 100d, GloVe 300d
[7]Downloadable from `https://code.google.com/archive/p/word2vec/` - Word2vec Pre-trained

2.3.2 Aggregating Word Vectors

Unweighted or simple average - One of the easiest and widely used ways to obtain a vector representation of a document from semantic word vectors is by averaging them. The average of semantic word vectors is obtained by averaging the components along each axis for all the words in the document. In a supervised setting, this representation is often used in combination with a neural network (De Boom et al., 2016). However, this thesis focuses on the unsupervised representation of documents. So let's take an example of how the document vector is generated from a simple average of word vectors. To start with, let us assume the 4-dimensional word vectors given below for each of the three words *I*, *love* and *reading* -

I	0.17	0.33	0.76	0.54
love	0.82	0.19	0.28	0.47
reading	0.51	0.42	0.16	0.74

(Example 9.1)

Given these word vectors, the sentence *I love reading* can be represented as a simple axis-wise average of components of all the word vectors. So, for example, the component along the first axis of the document vector will be the average of components along the first axis for each word in the document, i.e. $(0.17 + 0.82 + 0.51)/3 = 0.50$. Similarly, the complete document vector will be as given below -

I love reading	0.50	0.31	0.40	0.58

(Example 9.2)

The document vector so obtained represents the document as a vector in the same semantic space as the word vectors. This representation in Example 9.2 is a result of an unweighted average, i.e. each word vector is equally weighted in the averaging.

TF-iDF weighted average - To add some weight to the contribution of word vectors in the averaged vector that represents the document, a weighted averaging of the word vectors can also be applied. For example, the component-wise averaging of the word vectors could be weighted by the TF-iDF score of the word in the corpus. Let's say that the TF-iDF scores for the words *I*, *love*, *reading* are 0.1, 0.5 and 0.2, respectively. Then for a TF-iDF weighted average of word vectors, each word vector will be weighted by its TF-iDF score and then the weighted components are averaged for each axis. So, for example, the component along the first axis of the document vector will be the TF-iDF weighted average of components along the first axis for each word in the document, i.e. $(0.17 \times 0.1 + 0.82 \times 0.5 + 0.51 \times 0.2)/3 = 0.18$. The complete document vector as a result of TF-iDF weighted averaging of word vectors is given below -

I love reading	0.18	0.07	0.08	0.15

(Example 9.3)

Smooth Inverse Frequency weighted average - Another weighting scheme presented in Arora et al. (2017)[8], called *Smooth Inverse Frequency* (*SIF*), applies a smoothened inverse frequency weighted averaging on the word vectors. After weighted averaging, it applies an additional smoothening step of removing principal components. Therefore, the process of generating document vectors by the process described in Arora et al. (2017) is a two-step process.

First of all, the weighted averages of the word vectors are obtained. For averaging the components of word vectors each word vector is weighted according to the following weighting scheme -

$$f(w) = \frac{a}{(a + P(w))} \tag{2.6}$$

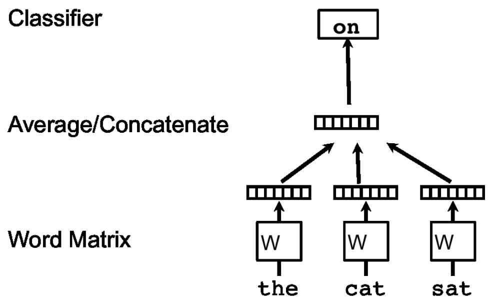
where, $f(w)$ is the weight for the word vector of word w, $P(w)$ is the probability of word w in the entire corpus and a is a term used for smoothening the inverse frequency[9]. The probability $P(w)$ is calculated based on the frequency of the word w in the dataset. The more frequent the word w will be in the corpus, the smaller will be its weight $f(w)$, and therefore its contribution in the averaged word vectors. This is expected to attenuate the contribution of some frequent words that appear frequently regardless of the document, like stopwords.

The second step in SIF methodology applies an additional smoothening step to the averaged word vector obtained in the previous step. Arora et al. (2017) observe that the averaged vector seemed to have huge components along semantically meaningless axes. Therefore, Arora et al. (2017) proposed an additional smoothening of the document vector which removes the projections of the averaged vector on the first singular vector. Arora et al. (2017) expect it to correct the document vector with respect to the most frequent text that is often related to syntax. The vector obtained after the correction is considered the final document vector by SIF methodology. Arora et al. (2017) report that Smooth Inverse Frequency weighted averaging outperforms the unweighted and TF-iDF weighted averaging of the word vectors in most of the Semantic Textual Similarity tasks (2012-2015) (Agirre et al. (2012); Agirre et al. (2013); Agirre et al. (2014); Agirre et al. (2015))

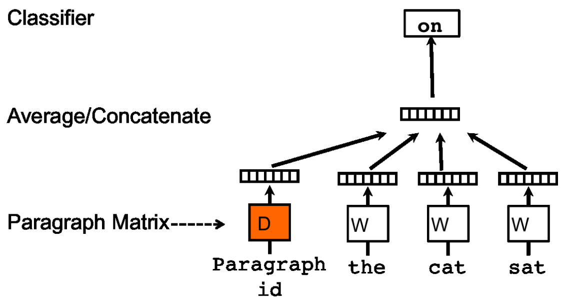
In this section, we have discussed various methods that employ weighted or unweighted averaging of semantic word vectors for representing documents. However, regardless of whether it is weighted or unweighted, these methods average the word vectors to represent documents in the form of a vector. The semantic word vectors place the words in a semantic vector space and the components play an important role in this placement. All the components across all the axes together place the word in this vector space. However, by averaging the components of the word vectors, we lose the placement of individual words and thereby, much of the meaning encoded in that placement of words. Therefore, while the averaging of semantic word vectors can represent documents, there may be better alternatives.

[8]GitHub repository - `https://github.com/PrincetonML/SIF`

[9]In Arora et al. (2017), a is called *weight parameter*.



(a) A framework for computing word vectors where the next word in the sequence is predicted on the basis of the concatenation or average of the preceding words in the sequence.



(b) A framework for computing the paragraph vector using distributed memory model where the the next word in the sequence is predicted on the basis of the concatenation or average of the preceding words in the sequence as well as a paragraph vector.

Figure 2.3: Analogy drawn by Le et al. (2014) between the framework used for computing word representations (Bengio et al., 2003) and document representations using Distributed Memory Architecture of Doc2vec.

2.4 Doc2vec

Inspired by the neural network models for word vectors, a similar architecture for learning document vectors was proposed by Le et al. (2014). Owing probably to the basis of inspiration behind the model and the subsequent resemblance with Word2vec, along with the model's objective of converting documents to vectors, the approach has come to be popularly known as *Doc2vec*. The authors have used the term "paragraph" to refer to a document that is possibly made up of more than one sentence, and therefore provide a model for computing paragraph vector (PV).

In the Doc2vec model, the word vectors are computed similar to Word2vec and the paragraph vector is computed along with the word vectors (Figure 2.3b). But unlike Word2vec, though the word vectors are shared across all the training samples, the paragraph vectors are shared only for the training samples from the same paragraph (as shown in Figure 2.3). Therefore, the word vectors are shared across the contexts sampled from all paragraphs, but the paragraph vectors are limited to the contexts sampled from the same paragraph. This allows for any difference in the prediction that is specific to the paragraph to be backpropagated and encoded in the paragraph vector. To this end, Le et al. (2014) defines two architectures - Distributed Memory (PV-DM) and Distributed Bag of Words (PV-DBOW).

The architecture referred to as the *Distributed Memory (PV-DM)* is similar to the sequential language modeling for word vectors in Bengio et al. (2003). In the neural network model proposed for words in Bengio et al. (2003), the next word in a sequence is predicted on the basis of concatenation/average of the word vectors of the preceding words in the sequence. In Distributed Memory (PV-DM) architecture as well, the next word in the sequence is predicted based on the concatenation/average of the word vectors of the preceding words in the sequence and an additional paragraph vector. This analogy drawn by the authors is given in Figure 2.3 using their original example. Unlike the Distributed Memory (PV-DM) architecture, the *Distributed Bag of Words (PV-DBOW)* architecture disregards the sequence of words. Despite its name PV-DBOW bears more similarity to Skip-gram Word2vec instead of CBOW Word2vec. Just like Skip-gram architecture for Word2vec, in Distributed Bag of Words (PV-DBOW) architecture, the context words are to be predicted given the paragraph vector instead. Overall, Doc2vec results in dense[10] vector representations of low dimensionality for the documents.

Doc2vec is the state-of-art for representing documents in an unsupervised way. In comparison to DT-VSM, Doc2vec is able to represent the documents in a vector space of lower dimension while encoding the similarity of documents in terms of proximity in the Doc2vec vector space. However, except for the similarity indicated by proximity, the Doc2vec vectors are difficult to interpret as there is no way, yet, to make sense of the axes of its vector space, i.e. the features of the document vectors (Kim et al., 2017). This reduces the room for further analysis and the possibility of adapting the representation of documents based on the task, which can be significant, especially when the task is to be performed in an unsupervised fashion. Ultimately, in comparison to words, documents are much more complex. Therefore, a vector representation of documents that can be easily interpreted without compromising on the performance or the quality of representation, is definitely worth investigating. Such a technique is discussed in the next chapter and lies at the center of discussion in this thesis.

[10]Quoting Jurafsky et al. (2021), dense vector implies that "instead of vector entries being sparse, mostly-zero counts or functions of counts, the values will be real-valued numbers that can be negative"

Chapter 3

Word Clusters based Document Embedding (WcDe)

In the previous chapter, we've discussed various methods that can be used to represent the documents. But as we saw in Example 2.1, in Document-Term Association Vector each word is represented by a separate axis and the only association between the words can be defined in terms of documents they appear in. But a word is more than that - it's a concept. It holds a meaning, rather, multiple meanings, within a language that can stand on its own irrespective of the document. Latent Semantic Analysis attempts to group similar words under an artificial concept based on their document co-occurrence and represent the document in terms of these artificial concepts. But a word is understood less by the documents it appears in and more by the immediate context it is used in (Socher, 2015). Moreover, there can be many words with very similar meanings and there can be many meanings to a single word. Therefore, some words are not completely unrelated, while sometimes, the same words are being used in a completely unrelated sense. These two major issues, termed *synonymy* and *polysemy* respectively, are discussed in Deerwester et al. (1990) regarding information need and information indexing. But these issues apply to the representation of any document in general as well. Latent Semantic Analysis approach discussed in Deerwester et al. (1990) attempts to address these issues. However, does the LSA representation of a document really address the issues of synonymy and polysemy? This lingering question will be raised again and be partly addressed in Chapter 5 (Section 5.3).

Word embeddings are trained over a large amount of data and result in a semantic vector representation of the words such that the closer the words are in the word embedding vector space, the more they are used in similar contexts, the more they share with respect to the concept they stand for. In other words, the proximity in this vector space is proportional to the degree of similarity of usage of the words in similar contexts, thereby, providing a measure of relatedness between the words. On the other hand, the contextualized word embeddings can provide different word vectors for the same word depending on the context it appears in. Therefore, we can have two representations for the same word based on the context it is used in, thereby separating unrelated senses of the same word. This begs the question, will the issues of synonymy and polysemy not benefit from using the semantic representation of word vectors while representing the documents? With that goal in mind, we can look at several techniques that have been employed to represent documents as a function of the semantic word vectors of the words it contains, as discussed in Chapter 2. Some of these unsupervised techniques represent the document as a weighted/unweighted average of the word vectors corresponding to the words in the document. However, in word vectors, the semantic representation of a word is defined in the form of its components along a fixed number of axes. The variation of components in the axis is what separates the contextual information for one word in the vector space from another. Let's consider the example below -

$$\begin{aligned}\overrightarrow{\mathbf{a}} &= 12\hat{x} + 12\hat{y}\\ \overrightarrow{\mathbf{b}} &= -12\hat{x} + -12\hat{y}\end{aligned}$$

(Example 10)

Example 10 above, shows two vectors $\overrightarrow{\mathbf{a}}$ and $\overrightarrow{\mathbf{b}}$. The averaging of the components of these vectors across each of its axes, $\hat{x}$ and $\hat{y}$, results in the vector situated at the origin. This example merely aims to lay an emphasis on what is lost in averaging the components across each of its axes. In the word vectors, where the components along the axes play an important role in situating the word in this semantic space, averaging is one way to lose much of the distinction that is created after heavy computations over billions of documents. Weighted averaging merely leans the average in favour of one concept over the other. The result, however, is a vector in the same space as the words, albeit adjusted. But a document is much more complex than a word.

Two different documents can be centered around the same topics even though they don't share any words except the stopwords. For example, consider the two documents[1] below -

Document A	Document B
The film sequel took three prizes - voted top animated feature film, top film comedy and top sequel.	*The romance thriller won all six of the movie awards for which it was nominated.*

(Example 11.1)

Both the documents above are discussing two movies that won awards. But except for the stopwords, they don't share any words that would indicate this similarity in Document-Term Association Vector of these documents. However, there are words in the documents that are similar in meaning, e.g. *film* and *movie*, *awards* and *prizes*. And then, there are words that are used in similar contexts and refer to the same concept. For example, *comedy*, *romance* and *thriller* all refer to the genres of movies. If we could sort the words into groups such that the words within the groups are similar

[1]Excerpts are originally from the BBC dataset - http://mlg.ucd.ie/datasets/bbc.html. However, these have been slightly modified for the sake of the example.

to each other and words from different groups are dissimilar to each other, we could analyze the similarity of documents with respect to these groups of words. Such grouping of any entity is called *clustering* of that entity. In this case, the entities are the words, i.e. we want a clustering of words. Using clustering we can get clusters of words that are similar to each other. These word clusters can now be used to analyze the similarity of documents with respect to these clusters instead of the individual words. In other words, it means that instead of the exact words, both the documents may exhibit a similarity in the word clusters they contain. For example, consider the word clusters below -

#	**Cluster Members**	**Common concept**
1	comedy, romance, thriller, horror, drama	Genre
2	movie, movies, film, films	Movie
3	prizes, prize, award, awards, merit, honour	Award
4	rain, snow, thunderstorm	Weather
5	one, two, three, four, six, 1, 5, 6	Numbers

(Example 11.2)

As we can see that both the documents in Example 11.1 contain different words from cluster 1, 2, 3 and 5 (Example 11.2). Therefore, it is worth investigating whether representing a document in terms of clusters of similar words can be effective for encoding a document in a document vector space.

Moreover, let's say we do represent the documents in terms of clusters of similar words instead of the words themselves. That is to say that the documents are be represented by various clusters of words, where each cluster contains words that are very similar in their meaning or usage. This way, similar words are mostly represented by the same cluster. Therefore, representing the documents in terms of clusters of similar words can provide a better solution to address the issue of synonymy. On the other hand, using contextual word embeddings different meanings of words can be clustered in different clusters. So, the clustering of words based on their contextualized semantic word representation can allow the same word with unrelated senses to be represented by different clusters, thereby, addressing polysemy. Therefore, the representation of documents in terms of clusters of similar words has the potential to address both issues. However, within the scope of this thesis, only the synonymy is addressed and the addressal of polysemy is a work for the future. The sections that follow will first outline the methodology, followed by a step-by-step example of the process and ending with a slightly more detailed discussion of each component of the methodology with respect to its role in this methodology.

3.1 Methodology

In order to represent a document in terms of a cluster of words, we need to first define the vector space for such a document vector. The vector space for such document vectors can be defined with these three major components -

Axes - Each axis represents a cluster of words. These word clusters are the same for all documents that we want to represent in a vector form.

Dimension - The size of the resulting vector will depend on the dimension of the vector space in which the document vector is represented. Since each axis represents a cluster of words, the dimension of this vector space indicates the number of word clusters that the document will be represented in terms of.

Components - Each document will have a component along each axis in this vector space. Each component represents the weight of the corresponding word cluster in that document.

Based on the definition of the vector space above, the methodology for generating document representations in terms of word clusters can be defined as -

Word Cluster based Document Embedding (WcDe) In this thesis, *Word Cluster based Document Embedding (WcDe)* refers to the document vectors that are represented in terms of word clusters obtained by clustering semantic word vectors. In a document vector generated by WcDe methodology, each axis represents a cluster of words and the corresponding component represents the weight of the entire cluster of words in the document.

(Definition 7)

The dimension and the axes are common for all the document vectors in WcDe vector space. However, the components along each axis are individual for each document. Therefore, the process of generating WcDe vectors happens in two parts -

Define Vector Space - Since both the dimension and axes that define the vector space depend on the word clusters, the first step is to cluster the words. In WcDe, the word clusters are obtained from clustering the semantic word vectors. For this purpose, semantic word vectors are clustered using clustering methods like K-Means, Agglomerative Hierarchical Clustering, etc. A few options for both and their respective effects on WcDe document representation will be discussed in detail in Sections 3.3.1 and 3.3.2.

Compute Document Vector of each Document - Once the word clusters have been obtained, each document is processed to attribute a weight corresponding to each word cluster. This weight of a word cluster acts as a component along the axis that represents the word cluster. So, if we have clustered the word vector space into, let's say, m clusters, a document can be represented as a vector $\overrightarrow{\mathbf{d}}$ which can be defined as an m-dimensional vector, as shown below -

$$\overrightarrow{\mathbf{d}} = \sum_{i=1}^{m} w_i \hat{q}_i \tag{3.1}$$

where, m is the number of word clusters, w_i is the weight of i^{th} cluster in the document, q_i is the i^{th} cluster and $\hat{q}_i$ is the unit vector along the axis that represents q_i. Different weighting schemes for WcDe that have been used in the past are discussed in detail in section 3.3.3. But for the sake of clarity right now, let's take a look at two simple weighting schemes based on - binary incidence and word cluster frequency. For example, let's say that we have to compute cluster weight for a cluster of words given by the set {*thriller*, *horror*, *comedy*} in the document[2] given below -

> *The original version of* <u>*horror*</u> *prequel Exorcist: The Beginning, which was dropped by the producers over claims that it was not scary enough, is to have its world premiere. Moreover, the low-budget* <u>*horror*</u> *film Boogeyman has knocked Robert de Niro* <u>*thriller*</u> *Hide and Seek from the top spot at the UK box office. In Hide and Seek, De Niro plays a widower whose daughter has a creepy imaginary friend. Despite lukewarm reviews from critics, the film took more than the expected \$18m (£9.5m). "The element of a real actor in a psychological* <u>*thriller*</u> *certainly elevated it," said Bruce Snyder, president of domestic distribution at 20th Century Fox.*
>
> *(Example 12)*

Then, a few simple ways to weigh the word clusters and calculate components for each axis can be -

Binary - According to the *binary* weighting scheme, if any word from a cluster is present in the document, then the weight for that word cluster is 1, else 0. So, according to this weighting scheme, the weight of cluster {*thriller*, *horror*, *comedy*} for the document given in Example 12 will be 1.

Cluster Frequency - It refers to the number of times any word from the cluster is present in the document. Therefore, for all the words that are a part of the cluster, the *cluster frequency* is the sum of frequencies of each of these words in the document. Therefore, according to this definition, the cluster frequency based weight of cluster {*thriller*, *horror*, *comedy*} for document given in Example 12 will be $2 + 2 = 4$.

3.2 An Example

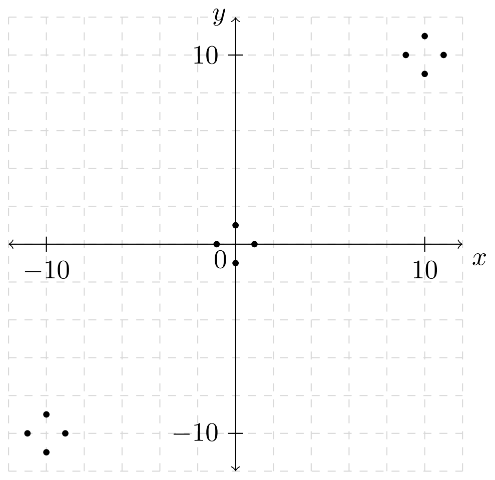
Before I discuss these steps in further technical detail, let's walk through a simple example. For the sake of fitting the example in and keeping it as simple as we can, let's limit the vocabulary of our example. Before we even take a look at the documents to be represented, let's assume that we want to represent our documents in terms of the word clusters of only 12 words given in Table 3.1. The actual vocabulary of our example dataset may be more than these 12 words or maybe less than these 12 words. But we are only concerned with these 12 words as far as the current document representation goes. In other words, any word that is not a part of these 12 words will not count towards representing the documents. In Table 3.1, I've also given arbitrary vector representations to each word for the sake of the example. The similar words are placed closer in the vector space

[2]The text was pieced together from multiple texts from the BBC dataset - http://mlg.ucd.ie/datasets/bbc.html

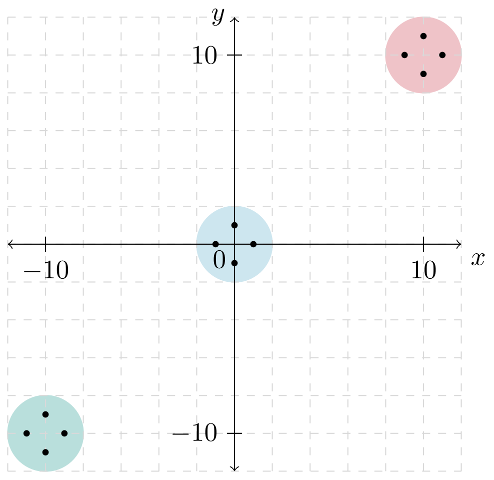
by design, to simulate the semantic word representations of the words. Figure 3.1a shows the word vectors in a 2-dimensional vector space to provide a visualization of their placement with respect to each other. Now that we have limited the vocabulary and defined some word vectors for the sake of the example, let's work on the example in detail below.

Word	**x**	**y**
cat	9	10
dog	11	10
pig	10	11
cow	10	9
car	-9	-10
bike	-11	-10
bus	-10	-11
cycle	-10	-9
green	0	1
red	1	0
yellow	0	-1
blue	-1	0

Table 3.1: Word vectors given for the step-by-step example of WcDe methodology



(a) A visualization of word vectors (indicated by black dot)



(b) A visualization of clusters of word vectors (indicated by colored dots enveloped in colored circles)

Figure 3.1: A visualization of word vectors given for the step-by-step example of WcDe methodology in 2-D (Table 3.1)

Defining Vector Space

As discussed in the Methodology above, the first step requires that we define the vector space for WcDe document vectors. In this preparatory task, we cluster word vectors to get word clusters.

As we can observe in Figure 3.1a, there are 3 clear clusters of word vectors. In the next step, we can employ a suitable clustering technique to get the word clusters. However, for the example, let's proceed with the assumption that we get the three clusters as shown in Figure 3.1b. In Figure 3.1b, each colorful circle (red, blue, green) that encapsulates the word vectors, indicates a word cluster. Based on the clusters shown in the figure, Table 3.2 shows 3 tables, each of which corresponds to a cluster and lists the word vectors that are a part of that cluster.

Word	**x**	**y**
cat	9	10
dog	11	10
pig	10	11
cow	10	9

(a) Cluster 1
(Red, Top Right)

Word	**x**	**y**
green	0	1
red	1	0
yellow	0	-1
blue	-1	0

(b) Cluster 2
(Blue, Center)

Word	**x**	**y**
car	-9	-10
bike	-11	-10
bus	-10	-11
cycle	-10	-9

(c) Cluster 3
(Green, Bottom Left)

Table 3.2: Word clusters based on the word vectors given in Table 3.1 for the step-by-step example of WcDe methodology

Each of these clusters form an axis in the document vector as shown below -

Cluster 1	Cluster 2	Cluster 3
$\hat{q}_1$	$\hat{q}_2$	$\hat{q}_3$

where $\hat{q}$ represents the unit vector along each axis that represents the word cluster, and the empty cells represent the component along each axis that is yet to be determined. This is the end of the preparatory step and the step that defines the WcDe vector space. At this stage in our example, we have a 3-dimensional document vector space with each axis representing a cluster shown in Table 3.2. In the next step, we will process each document to calculate weight for each word cluster for the document.

Computing document vectors

In order to represent documents in terms of word clusters, let's consider the 3 documents (sentences) given below -

Document 1	*I have a pig and a cow.*
Document 2	*There is a blue bike, a yellow cycle and a black bus.*
Document 3	*There is a black cat with a white dog in that red car.*

(Example 13.1)

For the sake of example, let us walk through the process for Document 1 only. In order to generate the WcDe document vector for Document 1, the document needs to be tokenized and split into tokens that can be matched against word clusters. So, let's consider the pre-processing steps given below -

Raw Document	*I have a pig and a cow.*
Casefolded	*i have a pig and a cow.*
Tokenized	*i, have, a, pig, and, a, cow, .*
Removed Special Characters	*i, have, a, pig, and, a, cow*

(Example 13.2)

Now, we will sort the tokens obtained after the pre-processing of Document 1 (Example 13.2), into the word clusters obtained in the previous step. After pre-processing, we can see that out of all the tokens only two are a part of our vocabulary of 12 words, i.e. *pig* and *cow*. Therefore, only these two will be sorted into clusters. Referring to Table 3.2, we can see that both of these tokens belong to Cluster 1 as shown in Example 13.3 below -

	Cluster 1 $\hat{q_1}$	Cluster 2 $\hat{q_2}$	Cluster 3 $\hat{q_3}$
d_1: *I have a pig and a cow.*	pig, cow		

(Example 13.3)

For the sake of simplicity, let's attribute a weight based on the cluster frequency in the document. As defined before, the cluster frequency refers to the total number of times any word from the word cluster appeared in the document. So for Document 1, based on the tokens sorted in clusters above (Example 13.3), the cluster frequency for each cluster is -

	Cluster 1 $\hat{q_1}$	Cluster 2 $\hat{q_2}$	Cluster 3 $\hat{q_3}$
d_1: *I have a pig and a cow.*	2	0	0

(Example 13.4)

The table above gives the WcDe document vector for Document 1. Just like we calculated cluster weights for Document 1 in the example above, we can calculate the cluster weights for all the documents from Example 13.1 as shown below -

		Cluster 1	Cluster 2	Cluster 3
		$\hat{q_1}$	$\hat{q_2}$	$\hat{q_3}$
d_1:	*I have a pig and a cow.*	2	0	0
d_2:	*There is a blue bike, a yellow cycle and a black bus.*	0	2	3
d_3:	*There is a black cat with a white dog in that red car.*	2	1	1

(Example 13.5)

Therefore, the WcDe vector of all the documents in Example 13.1 can be written in Cartesian representation as given below -

$$\overrightarrow{\mathbf{d_1}} = 2\hat{q_1} + 0\hat{q_2} + 0\hat{q_3}$$
$$\overrightarrow{\mathbf{d_2}} = 0\hat{q_1} + 2\hat{q_2} + 3\hat{q_3}$$
$$\overrightarrow{\mathbf{d_3}} = 2\hat{q_1} + 1\hat{q_2} + 1\hat{q_3}$$

(Example 13.6)

3.3 Discussion of Variations and Related Works

Based on the discussion so far, there are three fundamental components in the WcDe methodology -

1. The semantic word vectors or the word embedding
2. The clustering method used for clustering the semantic word vectors
3. The weighting scheme that attributes a numerical score to the relationship between the word cluster and the document

The first two define the WcDe vector space and the last one defines the component of each document vector along an axis in this vector space. For each of the three components of WcDe methodology, there are multiple options. In this section, I'll discuss a few options for each of these components.

3.3.1 Word embedding

The first step in the WcDe methodology is to define the WcDe vector space. In WcDe vector space, each axis represents a word cluster and the word clusters are obtained by clustering semantic word vectors. Therefore, in order to get the word clusters we first need the words to be represented in form of semantic word vectors such that the distance between the vectors indicates the similarity between words. In past research, many variations of the underlying methodology described in WcDe have been employed for various tasks. In each variation, there is often a different combination of the basic three components, but the underlying methodology remains the same. For example, though the underlying methodology was the same, but with respect to semantic word vectors, Kim et al. (2017), Qimin et al. (2015), and Mekala et al. (2017) trained word vectors using skip-gram Word2vec for their experiments, while Seifollahi et al. (2019) used GloVe to train word vectors from the datasets. On the other hand, instead of training word vectors, Dai et al. (2017) used pre-trained Word2vec word embedding for obtaining word vectors. Among the various options available for obtaining semantic word vectors, it is necessary to briefly discuss the impact of this decision with respect to generating WcDe document vectors.

However, before we discuss the impact of semantic word vectors, we must first discuss the impact of vocabulary. While clustering word vectors, we first need to identify the set of words that are to be clustered. When we train semantic word vectors from scratch from the dataset at hand, the word representations are learnt for each word in the vocabulary of the dataset. On the other hand, the pre-trained semantic word vectors have often been trained on a large amount of data (Table 2.1). Due to this, the vocabulary represented by the entire pre-trained word embedding often far exceeds the vocabulary of the dataset at hand. When the vocabulary of word embedding is limited to the vocabulary of the dataset at hand, the clustering will reflect word clusters based on the vocabulary of our dataset only. But if the vocabulary of the pre-trained embedding is not limited to that of the dataset, the word clusters will likely be more general rather than dataset-specific. But the vocabulary of the entire pre-trained word embedding contains noisy gibberish strings as well as words that are not relevant to the current dataset. This may lead to word clusters that do not serve any purpose for representing documents from our dataset. This can increase the dimensionality in the WcDe representation of documents and decrease the effectiveness as a representation. Therefore, we can cluster words beyond the vocabulary of our dataset, or limit them to what is relevant to our dataset our task. However, this decision can only be taken on a case-by-case basis. For most cases, the vocabulary of the dataset should suffice for the task. In this thesis, instead of the entire vocabulary of pre-trained word embedding, the experiments explore only the word clusters based on the vocabulary of the dataset at hand. Based on the vocabulary selected in this decision, we obtain the semantic vector representations for each word. However, there is more to consider. There are particularly two aspects of semantic word vectors that I'll center my discussion around -

1. Static vs Contextualized word embedding
2. Pre-trained vs Trained from scratch

Static vs Contextualized word embedding In static word embedding, a fixed representation is learnt for each word. However, in contextualized word embedding, the vector representation of a word is different in different contexts (Section 2.3.1). In other words, the word clusters obtained from the static word vectors will not account for different meanings of the same word. However, since WcDe methodology creates the word clusters based on the semantic word vectors, the use of contextualized semantic word vectors can lead to different senses of the same word being clustered in different clusters. This can address the difference in documents that, for example, refer to different senses of *book* (Example 6). However, this thesis analyses WcDe methodology with respect to static semantic word vectors only, and the use of contextualized word vectors is work for the future. With respect to word embedding, the focus of analysis in this thesis is more on the use of a word embedding trained from scratch versus a pre-trained word embedding that we'll discuss next.

Pre-trained vs Trained from scratch As we've discussed in Section 2.3.1, the semantic word vectors can be either trained on the dataset, or they can be obtained from word embeddings that have been intensively trained on a large amount of data. Since the pre-trained static semantic word vectors are computed over the use of a word in multiple contexts, it may fail to account for a word representation within a context that is specific to the dataset. That is, though the concepts represented by a word can be consistent, the vocabulary and its use might vary significantly depending on the intended audience and purpose of the communication. For example, the usage of the term *training* in the Machine Learning literature is closer to *computing*, while in fitness blogs, the term *training* refers to something else entirely. Therefore, training the word vectors on a selective dataset might provide the advantage of having word vectors computed through the contexts that are more pronounced and consistent in the dataset at hand. In this thesis, I will analyze the WcDe document vectors with respect to both types of semantic word vectors - pre-trained and the ones trained specifically on the dataset.

3.3.2 Clustering Techniques

The next step in the process of defining WcDe vector space is to define its axes. Since the axes in WcDe vector space represent word cluster, the next step is to cluster the word vectors obtained in the previous step. The choice of the clustering technique greatly impacts the performance, as we will see in the next chapter. However, let us briefly discuss a few clustering techniques that are relevant to the evaluation of multiple variations of WcDe that will be discussed in the next chapter. Clustering can be distinguished on mainly two fronts - the relationship among the clusters and the degree of participation of members in the clusters. With respect to the first front of distinction, the clustering that clusters data with no explicit relationship between clusters is called *flat clustering*, e.g. K-Means, whereas one that creates a hierarchy of clusters is known as *hierarchical clustering*, e.g. Agglomerative Hierarchical Clustering (Manning et al., 2008b). The second front that distinguishes clustering relates to full or partial membership of units (in this case words) in the cluster. If a unit can be a member of only one cluster, the clustering is called *hard clustering*. However, if the unit's membership can be distributed in clusters, i.e. the membership is neither binary nor limited to one cluster, then such clustering is known as *soft clustering* (Manning et al., 2008a). This thesis analyses WcDe methodologies with respect to flat and hierarchical hard clustering only. In this section I'll lightly discuss the two clustering techniques that have been used in the WcDe methodology in this thesis - K-Means and Agglomerative Hierarchical Clustering. Let's start with K-Means.

K-Means *K-Means* is a popular flat hard clustering technique. Let's say that we want to cluster some given points into k clusters. Then, given the points and a set of k initial cluster centers, also called *centroids*, K-Means clusters the points as given below -

1. **Assign each point** to its nearest cluster center.
2. **Recompute cluster centers** - New cluster center coordinates are given by a mean of all the points that are a member of the cluster.
3. **Repeat** steps 1-2 unless -
   - No points have changed clusters in step 1 since the last iteration, or
   - We've reached a pre-decided upper limit for the number of iterations, or
   - The objective function that we are trying to minimize falls below some pre-decided threshold. The objective function in the case of K-Means is called *residual sum of squares (RSS)*. RSS for K-Means is defined as the sum of squares of distance of each point from its assigned cluster center. It indicates how well the cluster center represents its members, and falling below the pre-decided threshold indicates that we've minimized the objective function enough (Manning et al., 2008a).

K-Means is simple and efficient clustering algorithm that is guaranteed to find local optimums (Manning et al., 2008a; Arthur et al., 2007). However, in some cases, it can optimize locally instead of globally and this significantly impacts the performance of clustering. That said, whether K-Means optimizes locally or globally can depend largely on the selection of initial cluster centers. And this is why the selection of initial cluster centers is crucial to the performance of K-Means. The initial cluster centers can be random points in the vector space, or they can be selected out of the set of points to be clustered. Instead of randomly selecting k points we can use the process of selection defined in Arthur et al. (2007). Arthur et al. (2007) define K-Means++ which provides a more efficient way to select the initial cluster centers. Please note here that K-Means++ improves only on the initial cluster selection of K-Means, the rest of the process remains the same as K-Means. In *K-Means++* the first cluster center is selected randomly from the given points that are to be clustered (Arthur et al., 2007). Thereafter, next cluster is selected based on a probability of its contribution in the objective function[3], i.e. RSS. By initializing the cluster centers so, K-Means++ significantly outperforms K-Means in terms of both - minimization of the objective function and the time taken for the algorithms to converge to an optimum (Arthur et al., 2007). Another step often taken to avoid local optimums in K-Means clustering, is to cluster multiple times with different initial cluster centers each time. In the end, the clustering with minimum RSS is selected. The K-Means clustering used in this thesis will use both K-Means++ initialization and multiple instantiations of K-Means in hopes to obtain global optimization of the clustering.

Agglomerative Hierarchical Clustering Hierarchical clustering is also another hard clustering technique. Unlike K-Means, which creates flat clusters, hierarchical clustering creates a hierarchy of clusters, as the name suggests. The approach to hierarchical clustering can be either top-down or bottom-up. The top-down hierarchical clustering starts with a cluster with all the points, and generates the hierarchy of clusters by dividing, and therefore is known as *divisive hierarchical clustering*. On the other hand, the bottom-up hierarchical clustering starts with single clusters and

[3]In Arthur et al., 2007, the objective function is referred to as *potential* of the clustering

agglomerates[4] as we go up and is therefore called *Agglomerative Hierarchical Clustering (AHC).* In Agglomerative Hierarchical Clustering, the clusters are merged on the basis of a *merge criterion* that we'll discuss soon. But for laying out the algorithm, let's assume that the clusters are merged on the basis of the cosine similarity of their centroids. Initially, in Agglomerative Hierarchical Clustering, each data point is considered a singleton cluster. Then, the similarity between each pair of singleton clusters is calculated and the rest of the process is as follows -

1. **Merge a pair of clusters** - In this step, the pair of clusters that is next according to the merge criteria is merged. Assuming that we are merging according to the cosine similarity of centroids, the pair with the highest cosine similarity, i.e. least cosine distance, is selected and merged.
2. **Calculate similarity of the merged cluster with other clusters** - In this step, we apply the merge criterion to the newly merged cluster. This adds the merged cluster at its rightful place in the next-to-be-merged list. Since we are assuming that the merge criteria is cosine similarity of centroids, in this step we calculate the similarity of the merged cluster with the rest of the clusters.
3. **Repeat** steps 1-2 until all clusters are merged.

This merging of clusters creates a hierarchy of clusters which is often depicted in a *dendrogram* as shown in Figure 3.2. Each horizontal line in the dendrogram shows a merge and its corresponding value at the y-axis indicates the cosine similarity value at which the clusters are merged as one. This similarity value where two clusters are merged is called the *combination similarity* of the merged cluster. For example, in Figure 3.2, the combination similarity for *horror* and *thriller* is 0.8. To get flat clusters from hierarchical clustering, we can cut the dendrogram at any value. So, if we want the minimum distance between the clusters to be, let's say 0.3, then that means that the clusters with a similarity of more than 0.3 should be merged. This cut-off value where we cut the dendrogram is also known as *distance threshold*. Therefore, as we can see in Figure 3.2, a distance threshold of 0.3 leads to 2 clusters - {*win*, *match*} and {*horror*, *thriller*, *comedy*}.

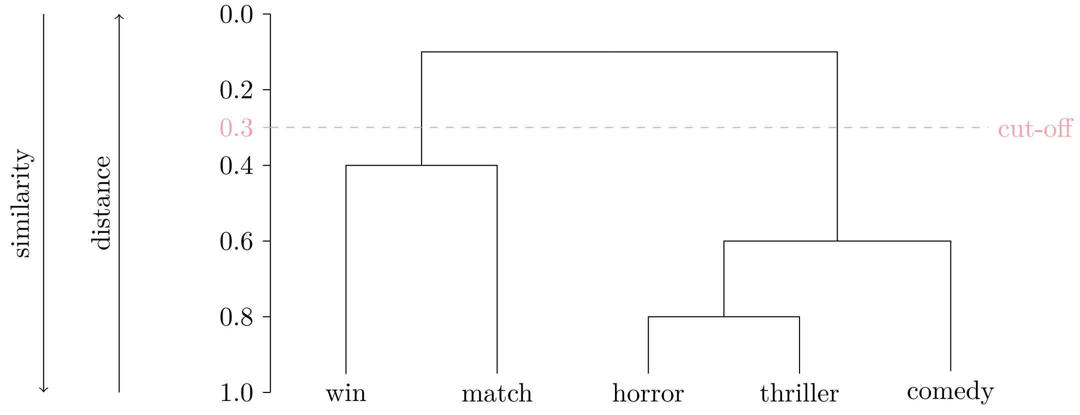
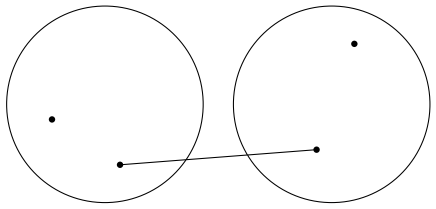
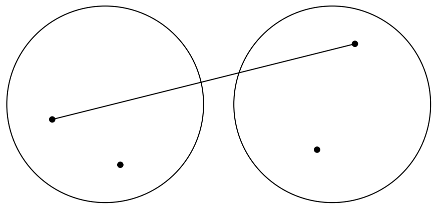


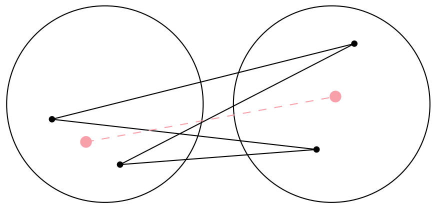
Figure 3.2: An example of a dendrogram of Agglomerative Hierarchical Clustering

[4] merges

As mentioned before, in order to merge clusters (step 1), we need to determine a merge criterion, i.e. a criterion that will allow us to decide which clusters to merge next. We started off by assuming that the merge criterion here refers to a similarity - particularly, cosine similarity of the centroids. However, this is just one way to define similarity between the clusters. There are several other ways. Figure 3.3 shows an example of two clusters with two members each. A solid line between two points indicates a *link*, which indicates that the similarity between these points defines the similarity of the clusters. In the case of multiple links, an average of the similarity of all the linked pairs defines the similarity of clusters. That said, the similarity is only one of such criteria, however, it is not the only criteria. Let's discuss a few merge criteria that can be used to select clusters to merge (Figure 3.3) -


(a) Maximum Similarity


(b) Minimum Similarity


(c) Centroid Similarity

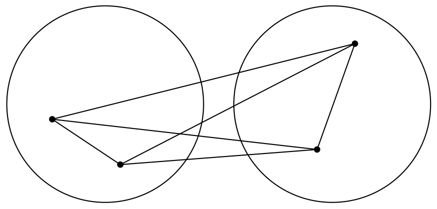

(d) Average Similarity

Figure 3.3: Various cluster similarity merge criteria for Agglomerative Hierarchical Clustering

Maximum similarity - The *maximum similarity* between two clusters means minimum distance between the clusters. This is calculated by selecting 1 point from each cluster such that the pair is closest to each other, i.e. the cosine similarity between the points is maximum and cosine distance is minimum (Figure 3.3a). In this case, the clusters are merged in the decreasing order of maximum similarity.

Minimum similarity - The *minimum similarity* between two clusters means the maximum distance between the clusters. This is calculated by selecting 1 point from each cluster such that the pair is farthest from each other, i.e. the cosine similarity between the points is minimum and the cosine distance is maximum (Figure 3.3b). In the minimum similarity criterion, the clusters are merged in the decreasing order of minimum similarity, i.e. the increasing order of maximum distance. So, the next pair of clusters to be merged will be the one where the distance between their farthest points is the least in comparison to other cluster pairs. In other words, the next pair of clusters to be merged will be the one where the similarity between their most dissimilar points is the highest in comparison to other cluster pairs. Therefore, the clusters that are merged in the decreasing order of their similarity are also the ones that will have the smallest radius when merged. That is why merging with minimum similarity will lead to compact clusters (Manning et al., 2008c). However, at the same time, it makes the clustering more sensitive to outliers as well.

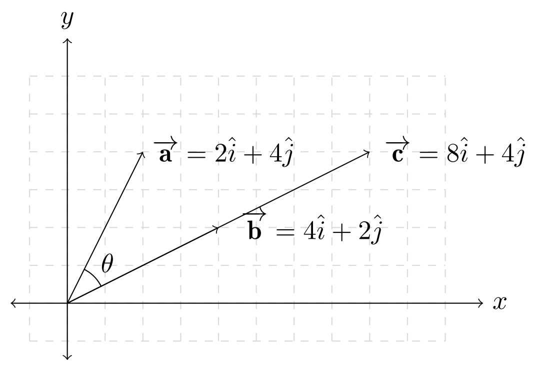
Centroid similarity - The *centroid similarity* between two clusters is given as the cosine similarity between the centroids of two clusters. In this criterion, the clusters are merged in the decreasing order of the similarity between their centroids. Figure 3.3c shows the example of 2 clusters with 2 points each. The centroids of these clusters are given by the red dots and the centroid similarity is calculated between the red dots (dashed line). But the centroids themselves are the average of the cluster members. So, mathematically, the centroid similarity can also be calculated as an average of similarities between all pairs of points, where each point in the pair belongs to a different cluster. This is especially useful in comparing the centroid similarity with average similarity, which is coming up next.

Average similarity - The *average similarity* between two clusters is the average of cosine similarity between **every** pair of the two clusters. Unlike centroid similarity, the cosine similarity between the pair of points belonging to the same cluster contributes to the average as well (Figure 3.3d). Just like the rest of the similarity criteria, with average similarity as criteria for merging, the clusters are merged in the decreasing order of their average similarity.

Minimum Variance (Ward's Method) - So far the merge criteria above have all selected the next pair of clusters to merge based on the similarity between the clusters. However, *Ward's Method* selects the next cluster based on the variance of the resulting merged cluster. The variance of a cluster is defined as the average of individual distances of all the points in the cluster from its cluster center. In terms of residual sum of squares (RSS)[5], variance is the average RSS of the merged cluster. Therefore, according to *minimum varianc* merge criteria, as the name suggests, the next pair to be merged, when merged, will have the minimum variance within the cluster in comparison to other pairs.

Related variations The selection of clustering techniques impacts the word clusters which are the foundation of the WcDe document vector space. I have discussed 2 clustering techniques in this section and I'll use both of them in the analysis in the upcoming chapters. However, in past various other clustering methods have been used to cluster word vectors leading to several variations of WcDe. Qimin et al. (2015) used K-Means for clustering the word vectors for the same underlying WcDe methodology. However, Kim et al. (2017) used spherical K-Means to cluster semantic word vectors. *Spherical K-Means* uses cosine distance instead of Euclidean distance in the calculation. However, cosine similarity may not be appropriate for measuring similarity between the semantic word vectors. Consider the following three points in a 2-dimensional vector space -

[5]object function of K-Means, discussed on page 41


y
x
$\overrightarrow{\mathbf{a}} = 2\hat{i} + 4\hat{j}$
$\overrightarrow{\mathbf{c}} = 8\hat{i} + 4\hat{j}$
$\overrightarrow{\mathbf{b}} = 4\hat{i} + 2\hat{j}$
θ

(Example 14)

In the above example, let $\hat{i}$ and $\hat{j}$ be the unit vectors along x and y axis respectively. As we can see from the figure, the cosine similarity between $\overrightarrow{\mathbf{a}}$ and $\overrightarrow{\mathbf{b}}$ is the same as cosine similarity between $\overrightarrow{\mathbf{a}}$ and $\overrightarrow{\mathbf{c}}$, i.e. $\cos\theta$. In semantic word vectors, the value of components along the axis plays an important role in placing the word in the semantic vector space. However, cosine similarity only takes the angular similarity into account. Therefore, cosine similarity may not be the best measure of similarity between semantic word vectors. Seifollahi et al. (2019), also used spherical K-Means, albeit adapted in a divisive hierarchical fashion, to cluster the word vectors into a hierarchy of clusters. The divisive clustering was implemented such that a cluster is split into two if its membership exceeds a set threshold.

Dai et al. (2017), on the other hand, adapted a flat hard clustering algorithm from the *Chinese Restaurant Problem* of Dirichlet Process to cluster the word vectors (Aldous, 1985). In their adapted clustering algorithm, every word that is processed for clustering has two choices - either be added to an existing cluster or create a new cluster. The probability of creating a new cluster is $1/(n+1)$, where n is the number of clusters. Therefore, the first word creates a new cluster and thereafter the probability of creating a new cluster only decreases. The process adapted by Dai et al. (2017) to cluster the words (after the first one) is as follows -

1. **Select next word** to process
2. **Generate random probability value** - Based on this random probability value, a word is either added to an existing cluster or to a new cluster
   i. **If** random probability is less than the probability of creating a new cluster, **then** a new cluster is created.
   ii. **Else** the new word is added to an existing cluster with which it has the maximum cosine similarity.
3. **Repeat** steps 1-2 for the next word until all the words are clustered.

According to the algorithm given in Dai et al. (2017), one part of the clustering is not even considering the semantic word vector (condition i.) when generating new clusters. Rather, a new cluster is generated based on a random value between 0 to 1. The part that does consider the semantic word vector is the one where a word is added to an existing cluster (condition ii.). However, this part (condition ii.) uses cosine similarity. As we've discussed, the cosine may not be the best distance measure to be used for comparing semantic word vectors (Example 14). Moreover, the algorithm works in a way that discourages the formation of new clusters. Therefore, the words are more likely to be added to an existing cluster. This leads to the formation of a few clusters, but at the same time, it leads to a skewed distribution of words in clusters. That is, some clusters have most of the members while many have barely a few members. Thus the larger clusters are likely to contain a lot of words and may not represent a coherence with respect to word similarity.

Unlike the works mentioned so far that use hard clustering, Mekala et al. (2017) applied soft clustering on word vectors using Gaussian Mixture Models (Reynolds, 2015). Therefore, in Mekala et al. (2017) a word could be a member of more than one cluster and to varying degrees. However, the discussion of Gaussian Mixture Models is beyond the scope of this thesis. To conclude, though there are various avenues of investigation of WcDe with respect to clustering algorithm, in this thesis, I am limiting my analysis of WcDe with respect to flat hard clustering techniques, specifically, K-Means and Agglomerative Hierarchical Clustering.

3.3.3 Weight Functions

The two steps above define the axes that form the vector space. Having defined the WcDe vector space, each document can now be placed in this vector space. Therefore, in this step, each document is processed to obtain a component for its document vector along each axis. This calculation of components is the next and the last step in the process of generating WcDe vectors. In the Methodology section (Section 3.1), I discussed two simple *weighting schemes* - binary and cluster frequency (Example 12). In *binary* weighting scheme, the *weight* or score for a word cluster in a document is 1, if any word from the cluster appears in the document. Similarly, in the weighting scheme based on *cluster frequency*, the score is the count of the number of times the document contained any word from the cluster. According to another weighting scheme used in Seifollahi et al. (2019), the weight of a word cluster can be calculated as a simple sum of TF-iDF score for each word in the document that belongs to the cluster.

However, many researchers have adapted the underlying modeling of TF-iDF score for Word Cluster based Document Embedding. The adapted function adapts the concepts of term frequency and inverse document frequency of the word in the form of cluster frequency and inverse document frequency of word cluster, respectively. The cluster frequency is as described above and *document frequency* of word clusters refers to the number of documents that contained at least one word from the word cluster. For example, let's consider the same word cluster that was used in Example 12, i.e. $\{$*thriller*, *horror*, *comedy*$\}$. Let's also assume that the entire corpus contains only the following three documents[6] for the sake of the example -

[6]Excerpts have been taken from the BBC dataset - http://mlg.ucd.ie/datasets/bbc.html

Document A	Document B	Document C
The original version of horror prequel Exorcist: The Beginning, which was dropped by the producers over claims that it was not scary enough, is to have its world premiere.	*Moreover, the low-budget horror film Boogeyman has knocked Robert de Niro thriller Hide and Seek from the top spot at the UK box office.*	*Preview performances of The Glass Menagerie will begin at New York's Ethel Barrymore Theatre on Thursday.*

(Example 15.1)

In Example 15.1 above, the document frequency of word cluster {*thriller*, *horror*, *comedy*} is 2 since no word from this word cluster appears in Document C. Similarly the cluster frequency for Document A, B and C is 1, 2 and 0, respectively. Just like TF-iDF, now these cluster frequencies and the document frequency of the word cluster can be used to compute, as Kim et al. (2017) called it, the *Cluster Frequency-Inverse Document Frequency (CF-iDF)* score of the word cluster.

Cluster Frequency-Inverse Document Frequency (CF-iDF) It is a weighting scheme adapted from the TF-iDF weighting scheme that, given a set of documents, calculates the relevance of a word cluster in a document. In the Cluster Frequency-Inverse Document Frequency weighting scheme, the word cluster and document association score is formulated in terms of cluster frequency (in the document) and inverse document frequency of the word cluster (in the entire set of documents).

(Definition 8)

A few sections ago, in Equation (3.1), a WcDe document vector $\overrightarrow{\mathbf{d}}$ was defined as an m-dimensional vector, as shown below -

$$\overrightarrow{\mathbf{d}} = \sum_{i=1}^{m} w_i \hat{q}_i$$

where, m is the number of word clusters, w_i is the weight of i^{th} cluster in the document, q_i is the i^{th} cluster and $\hat{q}_i$ is the unit vector along the axis that represents q_i. So based on the Equation (3.1), let's indicate the weight of i^{th} cluster, i.e. q_i, in the document d by $w_{i,d}$. Then the cluster weight $w_{i,d}$ in terms of CF-iDF score can be given as -

$$w_{i,d} = \text{cf-idf}_{q_i,d}$$

where, $\text{cf-idf}_{q_i,d}$ is the CF-iDF score of cluster q_i in document d. Now, similar to TF-iDF, the CF-iDF score can be formulated as -

$$\begin{aligned} w_{i,d} &= \text{cf-idf}_{q_i,d} \\ &= \text{cf}_{q_i,d} \times \text{idf}_{q_i} \\ &= \text{cf}_{q_i,d} \times \log\left(\frac{N}{\text{df}_{q_i}}\right) \end{aligned} \tag{3.2}$$

where $\text{cf}_{q_i,d}$ represents the frequency of the cluster q_i in document d and df_{q_i} represents the document frequency of word cluster q_i. Here, the occurrence of a word cluster in the document can be understood to be the same as the occurrence of any word from the word cluster in the document. Therefore, the document frequency df_{q_i} of cluster q_i is basically the count of documents where any word from the word cluster q_i appears. Similarly, the frequency of word cluster q_i in document d, i.e. $\text{cf}_{q_i,d}$, is the count of occurrences of any word from the cluster q_i in the document d. Based on this understanding, the cluster frequency $\text{cf}_{q_i,d}$ can also be given in terms of term frequency $\text{tf}_{t,d}$ as given below -

$$\text{cf}_{q_i,d} = \sum_{t \in q_i} \text{tf}_{t,d} \tag{3.3}$$

To further understand the formulation of CF-iDF, let's continuing working the Example 15.1 and calculate the CF-iDF score of the same word cluster $q_x = \{\textit{thriller}, \textit{horror}, \textit{comedy}\}$ in each document that is given in the example. Since we are considering a corpus of only these three documents, i.e. A, B, and C, the total number of documents $N = 3$. The cluster frequency (cf), document frequency of the word cluster (df), and the CF-iDF score (cf-idf) for each document is as given below in Example 15.2. Please note that in the example, for the calculations involving logarithm, I have used the decimal logarithm ($\log_{10}$) as indicated in the table -

		$\text{cf}_{q_x,d}$	df_{q_x}	$\log_{10}\left(\frac{N}{\text{df}_{q_x}}\right)$	$\text{cf-idf}_{q_x,d}$
d_A:	*The original version of horror prequel Exorcist: The Beginning, which was dropped by the producers over claims that it was not scary enough, is to have its world premiere.*	1	2	0.176	0.176
d_B:	*Moreover, the low-budget horror film Boogeyman has knocked Robert de Niro thriller Hide and Seek from the top spot at the UK box office.*	2	2	0.176	0.352
d_C:	*Preview performances of The Glass Menagerie will begin at New York's Ethel Barrymore Theatre on Thursday.*	0	2	0.176	0

(Example 15.2)

In Qimin et al. (2015), after computing the weights for each feature, the document vector was length normalized as given in Equation (3.4). The length normalized form of document vector from Equation (3.1) is given below as -

$$\begin{aligned}\overrightarrow{\mathbf{d}} &= \sum_{i=1}^{m} \left(\frac{w_i}{|\overrightarrow{\mathbf{d}}|} \right) \hat{qi} \\ &= \sum_{i=1}^{m} \left(\frac{w_i}{\sqrt{\sum_{j=1}^{m} w_j^2}} \right) \hat{qi} \end{aligned} \tag{3.4}$$

In this section we've seen that the WcDe methodology can be implemented with a variety of options at each step. At each step an option can be selected depending on the problem. In the upcoming chapters, I will not only compare this methodology against the other document representation methods discussed in Chapter 2, but also among its several variations. Moreover, we will further explore, evaluate and discuss this methodology and the subsequent results in detail in Chapters 4 and 5.

Chapter 4

Evaluation & Quantitative Analysis

In the previous chapters, we've discussed various methods to represent the documents. But, how do we evaluate these techniques? How do we determine if one document vector is a better representation of the document than the other? In the beginning of this chapter, we will first create the foundation of our evaluation and this will pave way for the results that will be discussed and analyzed thereafter.

So, what should be the criteria for determining which document representation is better in comparison? The answer to that depends on what it is that we expect the document vector to represent. When we represent a word in a semantic vector space, it is expected that similar words should reflect that similarity in the vector space in terms of proximity. Using an analogous hypothesis for documents, we can say that similar documents should be closer in the vector space as well. However, unlike words, the similarity of documents can be much more subjective. Moreover, the larger the document, the more arguable and subjective the notion of similarity between two documents becomes. But the takeaway for evaluation is that a better representation of the documents in the vector space should better map their relative similarities and dissimilarities in the document vector space. The similarity itself can be at different levels - all the way from being identical to only sharing a theme. For the evaluation in this thesis, the similarity between the documents is assessed only at a topical level.

Therefore, we expect that the distance between the document vectors should reflect the relative topical similarity. That is, the document vectors are expected to be closer in vector space if they belong to the same topic. But the difference between some topics can be more apparent than the others. For example, when comparing among a set of topics, some topics can be quite different like Sports and Business, while some can have a common underlying theme like Rugby and Football, which are both related to Sports (Figure 4.1). Therefore, even while evaluating document representations on topical similarity, we can evaluate the document vectors at different levels of topical similarity. This allows us to evaluate the document representations in terms of different levels of topical similarity. This also demonstrates how well the representation can map the difference between the documents that differ less due to the presence of an underlying common theme. Therefore, in my experiments, I've evaluated the document representations on specifically two levels of topical similarity by using BBC datasets.

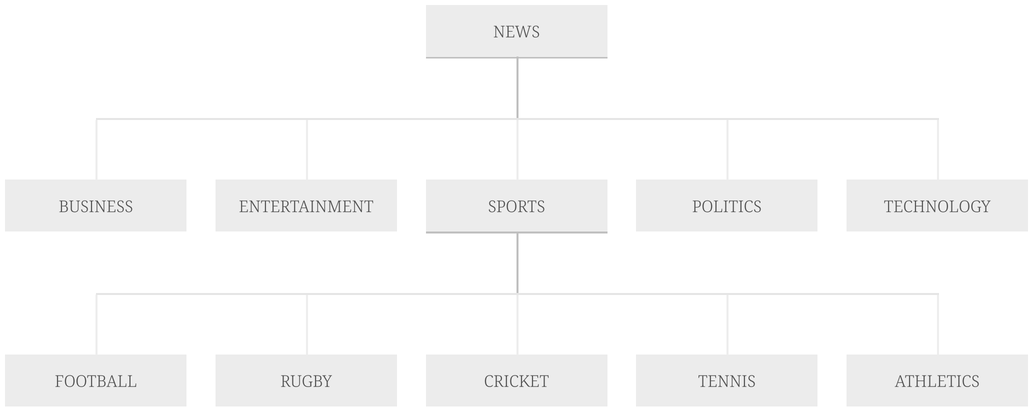


Figure 4.1: Hierarchy of the topics/classes/categories in the BBC datasets

4.1 Dataset & Task

The BBC Datasets[1], one of the many datasets that can used as a benchmark for machine learning research, consist of two datasets - BBC and BBC Sport. Both the datasets contain news articles spanning a total of 5 topical areas each (Figure 4.1). The difference between the datasets, however, lies in the categories. While the former contains general news articles (Table 4.1a), the latter consists of specifically `sports` news articles spanning 5 sub-categories of sports news (Table 4.1b).

Class	Count
business	510
entertainment	386
politics	417
sport	511
tech	401
Total	**2225**

(a) BBC

Class	Count
athletics	101
cricket	124
football	265
rugby	147
tennis	100
Total	**737**

(b) BBC Sport

Table 4.1: Class-wise frequency distribution of BBC datasets

Since these datasets contain two levels of topic distribution, these datasets can be used for evaluating the performance of topical similarity of documents at different levels. Moreover, the role of representation is more evident in unsupervised tasks. Therefore, to compare the representations all by themselves, in my experiments, I've used clustering to ascertain the similarity of the document vectors with respect to the topic of the document's content.

[1]BBC Datasets are available at http://mlg.ucd.ie/datasets/bbc.html

In *document clustering*, documents are grouped into a set of clusters such that there is strong -

- ***Intra-cluster similarity*** - The documents belonging to the same cluster are very similar
- ***Inter-cluster dissimilarity*** - The documents belonging to the different cluster are very dissimilar

(Definition 9)

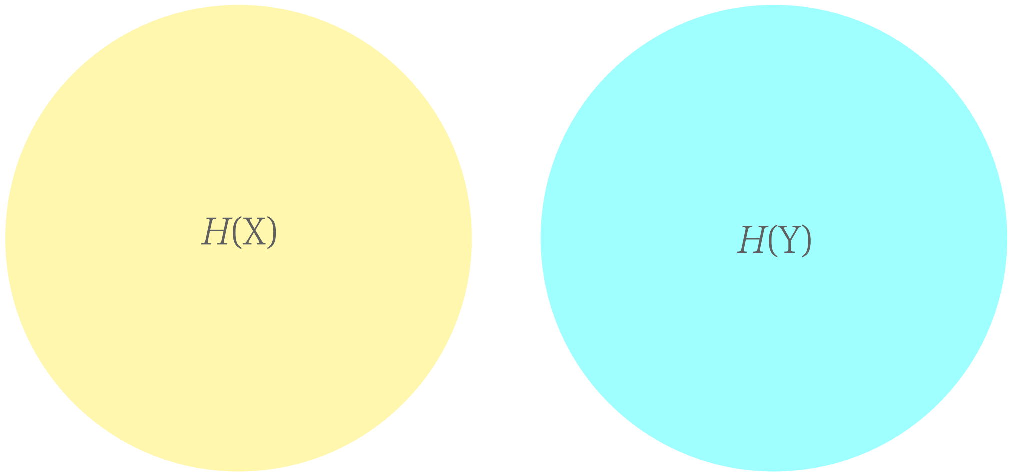
In the experiments, each document in the dataset is represented in the form of a vector using various techniques discussed so far and then clustered into as many clusters as there are classes in the dataset using K-Means clustering technique. The quality of clustering is then evaluated by comparing the clusters with the categories, i.e. *true class*, of the documents using different metrics. Before we dive into the results and exact parameters of the experiment, we first need to discuss the measures for comparing the performance. In the following section, we'll discuss a few evaluation metrics that were used for quantitative comparisons.

4.2 Evaluation Metrics

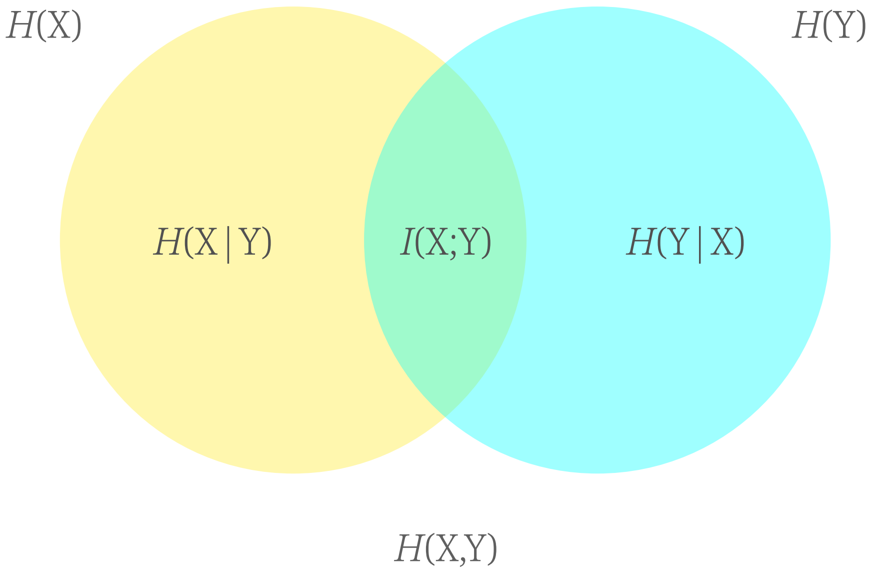
Clustering is an unsupervised approach. Often when we cluster data, we do not know the true class a document belongs to. Therefore, clustering can be usually evaluated with or without the knowledge of true class labels. The approach to evaluation where the quality of clustering is determined based on the inter-cluster and intra-cluster similarity and dissimilarity only, without the knowledge of true labels or true class of documents, is called *internal evaluation criterion.* The other approach, where we use the *external* judgement, true label for example, as a benchmark and evaluate on the basis of how well it matches the clustering solution, is called *external evaluation criterion.* Among the metrics that follow the external evaluation criterion, some rely on the step after clustering where each cluster is assigned a class based on some parameter, for example, Purity, F-Measure. These metrics evaluate the performance of clustering by mostly measuring the agreement between the assigned cluster label and the true class label. Some other metrics evaluate on the basis of information theory and analyse the distribution of class and cluster membership, for example, Entropy, Mutual Information etc. Another set of metrics use a combinatorial approach where the performance is measured in terms of keep-together or keep-separate decisions for each possible pair. A few examples of such evaluation metrics are Rand Index and Fowlkes Measure (Rosenberg et al., 2007). In this thesis, I'll use Normalized Mutual Information, Adjusted Rand Index and Fowlkes Mallows Index to evaluate the clustering. These evaluation metrics are discussed in the sections that follow.

4.2.1 Mutual Information

Mutual information in clustering can be used to measure the agreement between two sets of labels. Assuming a fundamental understanding of entropy and conditional entropy, let's take a look at the Figure 4.2a to understand Normalized Mutual Information as a clustering evaluation metric (Cover



(a) Individual entropies $H(\mathrm{X})$ of X (left, yellow) and $H(\mathrm{Y})$ of Y (right, blue)



(b) A Diagram analogous to Venn Diagrams for sets showing additive and subtractive relationships of various information measures associated with the correlated variables X and Y (Cover et al., 2005). The total area contained by both circles (yellow, blue and green portions) is the joint entropy $H(\mathrm{X},\mathrm{Y})$. The left circle (including both yellow and green portions) represents the individual entropy $H(\mathrm{X})$ of X, while the non-intersecting part of left circle (yellow portion) indicates the conditional entropy $H(\mathrm{X}|\mathrm{Y})$. Similarly, the right circle (including both green and blue portions) represents the individual entropy $H(\mathrm{Y})$ of Y, while the non-intersecting part of right circle (blue portion) indicates the conditional entropy $H(\mathrm{Y}|\mathrm{X})$. The green portion that forms the intersection of $H(\mathrm{X})$ and $H(\mathrm{Y})$ is the mutual information $I(\mathrm{X};\mathrm{Y})$.

Figure 4.2

et al., 2005). Figure 4.2a shows the individual entropies of random variables X and Y whereas Figure 4.2b figure shows the additive and subtractive relationships of various information measures associated with correlated variables X and Y. In Figure 4.2a, the entire left yellow circle represents the entropy of X, i.e. $H(\mathrm{X})$ and similarly, the entire right blue circle represents the entropy of Y, i.e. $H(\mathrm{Y})$. In Figure 4.2b, the green portion where both the entropies (given by circles) overlap is the mutual information $I(\mathrm{X};\mathrm{Y})$. As we can infer from the figure, the Mutual Information between X and Y can be equivalently expressed in terms of entropies below -

$$
\begin{aligned}
I(\mathrm{X};\mathrm{Y}) &= H(\mathrm{Y}) - H(\mathrm{Y}|\mathrm{X}) && (4.1)\\
&= H(\mathrm{X}) - H(\mathrm{X}|\mathrm{Y}) && (4.2)\\
&= H(\mathrm{X},\mathrm{Y}) - H(\mathrm{Y}|\mathrm{X}) - H(\mathrm{X}|\mathrm{Y})
\end{aligned}
$$

Let's take a look at each component at the right-hand side of Equation (4.1) intuitively[2]. If entropy $H(\mathrm{Y})$ is regarded as a measure of uncertainty about a random variable Y, then the conditional entropy $H(\mathrm{Y}|\mathrm{X})$ is a measure of uncertainty about a random variable Y despite knowing X. Therefore the difference between total uncertainty and uncertainty despite knowing X is the uncertainty reduced because X was known. In other words, the difference of the two quantities can also be interpreted as information gained about Y because X was known. Similarly, looking at Equation (4.2), we can understand that the difference of two quantities on the right hand side indicates the information gained about X because Y was known. As we can see, this *Information Gain* is mutual and therefore corroborates with the term Mutual Information.

In the evaluation of clustering, the Mutual Information score is popularly used to measure the agreement between two assignments. So far, we've looked at it in terms of entropy. However, it can also be formulated in terms of probabilities and set cardinalities. Let N be the total number of documents, $\mathbb{C} = \{c_1, c_2, c_3, ..., c_m\}$ be the set of classes and $\Omega = \{\omega_1, \omega_2, \omega_3, ..., \omega_n\}$ be the set of clusters where, c_i and ω_j are the sets of all documents belonging to class i and cluster j, respectively. Equation (4.3) formulates the Mutual Information between the true class labels $\mathbb{C}$ and cluster assignments Ω with the probabilities instead of entropies. For evaluation of clustering, however, the probabilities in Equation (4.3) can be further estimated in terms of set cardinalities using maximum likelihood estimates[3] as shown in Equation (4.4) (Manning et al., 2008a) -

$$
\begin{aligned}
I(\Omega;\mathbb{C}) &= \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)} && (4.3)\\
&= \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \log \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|} && (4.4)
\end{aligned}
$$

Normalized Mutual Information (NMI) The Mutual information score is the maximum for a clustering that perfectly matches the true classes. But it also increases with the number of clusters. In particular, the Mutual Information score is also maximum if the number of clusters is of the

[2]intuition borrowed from Wikipedia contributors, 2021.

[3]each probability is estimated as the corresponding relative frequency (Manning et al., 2008a)

same order of magnitude as the number of data points, i.e., $|\Omega| = N$. To address this, the Mutual Information can be normalized by using a mean of the entropies of each clustering as given in Equation (4.5) (Manning et al., 2008a).

$$\text{NMI}(\Omega, \mathbb{C}) = \frac{I(\Omega; \mathbb{C})}{mean(H(\Omega), H(\mathbb{C}))} \tag{4.5}$$

$$= \frac{I(\Omega; \mathbb{C})}{[H(\Omega) + H(\mathbb{C})]/2} \tag{4.6}$$

This is called the *Normalized Mutual Information*. Normalizing Mutual Information score by the arithmetic mean of entropies ensures that NMI is always between 0 and 1. Moreover, since entropy tends to increase with the number of clusters, it also penalizes the NMI for a large number of clusters.

In this thesis, Normalized Mutual Information is the primary metric for evaluation because it encompasses two aspects of clustering. For a good clustering, especially given true labels, there are at least two essential aspects of evaluation - that both the cluster and the class distribution be highly biased and skewed. The distribution of classes in a cluster being skewed in favour of one class indicates *homogeneity* of the cluster. Therefore homogeneity indicates both - the degree of presence of the majority class as well as the absence of other classes in the cluster. Similarly, the distribution of clusters in a class being skewed in the favour of a single cluster indicates how well a cluster envelops the entire class, i.e. *completeness* of the cluster. In other words, if the distribution of clusters is not skewed in the favour of a single cluster, then it indicates how spread out the class is among all the clusters, and how incomplete the cluster is. Rosenberg et al., 2007 defines both of these aspects - homogeneity and completeness - in terms of entropy (Equations (4.7) and (4.8)).

$$\mathbf{h}(\Omega, \mathbb{C}) = \begin{cases} 1, & \text{if } H(\mathbb{C}, \Omega) = 0 \\ 1 - \frac{H(\mathbb{C}|\Omega)}{H(\mathbb{C})}, & \text{otherwise} \end{cases} \tag{4.7}$$

$$\mathbf{c}(\Omega, \mathbb{C}) = \begin{cases} 1, & \text{if } H(\mathbb{C}, \Omega) = 0 \\ 1 - \frac{H(\Omega|\mathbb{C})}{H(\Omega)}, & \text{otherwise} \end{cases} \tag{4.8}$$

where $\mathbf{h}(\Omega, \mathbb{C})$ is the Homogeneity of clusters and $\mathbf{c}(\Omega, \mathbb{C})$ is the Completeness of clusters. Mathematically, Mutual Information normalized with arithmetic mean is equivalent to the harmonic mean of both Homogeneity and Completeness (Becker, 2011). Since it reflects a balance of both Homogeneity and Completeness, for the experiments in this thesis, Normalized Mutual Information with arithmetic mean of entropies has been used as the primary evaluation metric.

4.2.2 Rand Index

Rand Index, is one of the clustering evaluation metrics that take a combinatorial approach, i.e. assess performance with respect to all possible pairs of data points. In simple words, it measures the fraction of all the pairs, that belong to the same class and have been correctly placed together in some cluster, and the pairs that belong to different classes and have been correctly placed in separate clusters.

Therefore, following the definition given above, for N data points, the Rand Index can be defined as given below -

$$\text{RI} = \left(\frac{x+y}{{}^{N}C_2}\right) \tag{4.9}$$

where, ${}^{N}C_2$ is the notation for "N choose 2". It gives the total number of unique combinations that we'll get when we select only 2 out of N at a time, i.e. the total number of pairs of documents. x is the number of pairs that have been clustered together, and in fact, also belong to the same true class; y is the number of pairs that have been assigned different clusters, and in fact, also belong to the different true classes. Since the Rand Index is the fraction of correct pair assignments out of all the total possible pair assignments, this can also be considered an accuracy measure for clustering evaluation (Manning et al., 2008a).

Another way to look at it is as the assignment of pairs to the same cluster or different clusters. So, the assignment of a pair of similar data points to the same cluster is considered a True Positive (TP) assignment, and similarly, the assignment of the dissimilar data points to different clusters is considered as True Negative (TN) assignment. Therefore, False Positive (FP) assignments will be of the pairs that were supposed to be assigned to different clusters but were assigned the same cluster instead. Similarly, the False Negative (FN) assignments will be of the pairs that were supposed to be assigned to the same cluster but were assigned different clusters instead. Based on the definitions given above, the Rand Index can also be defined as -

$$\text{RI} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{4.10}$$

$$= \frac{\text{TP} + \text{TN}}{{}^{N}C_2} \tag{4.11}$$

As we can see above in Equation in (4.11), x from Equation (4.9), essentially represents True Positive assignments and y essentially represents True Negative Assignments.

Adjusted Rand Index Both the Rand index and Normalized Mutual Information will hardly ever be close to zero. Even with randomly assigned cluster labels both the evaluation metrics may produce a low score but it is nowhere close to zero. Though both the scores are defined to always be between 0 and 1, there isn't an appropriate selection of clustering that will ever result in a score of 0. Hubert et al. (1985) defines an "corrected for chance" adjustment to Rand Index such that there is a constant value (e.g. 0) under an appropriate null hypothesis -

$$\text{ARI} = \frac{\text{RI} - E[RI]}{max(\text{RI}) - E[RI]} \tag{4.12}$$

$$= \frac{\text{RI} - E[RI]}{1 - E[RI]}$$

where, $max(\text{RI})$ is the maximum Rand Index, i.e. 1, and $E[RI]$ is the expected value of the Rand Index. This way the Rand Index takes on the value 0 when the value of the Rand Index is equal to the expected value of the Rand Index. This is called the *Adjusted Rand Index*. Since Adjusted Rand Index is akin to accuracy in clustering evaluation, it is used as an accuracy measure of the document clustering in this thesis.

4.2.3 Fowlkes Mallows Index

Just like Rand Index, *Fowlkes Mallows Index (FMI)*, also takes a combinatorial approach. In terms of the $\text{TP}, \text{TN}, \text{FP}, \text{FN}$ assignments defined in Adjusted Rand Index, Fowlkes et al. (1983) defined Fowlkes Mallows Index as -

$$\texttt{FM} = \frac{\text{TP}}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN})}} \tag{4.13}$$

where $\texttt{FM}$ refers to the Fowlkes Mallows Index. However, from Equation (4.13), we can derive Fowlkes Mallows Index in terms of two more relatable measures. As we discussed in (Section 4.2.2), Rand Index gives the Accuracy of pairwise assignments. Similarly we can also define the Precision and Recall in terms of the pairwise assignments as given below -

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.14}$$

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.15}$$

where P is the precision and R is the recall of the pairwise assignments. Based on the definition of Precision and Recall in Equations (4.14) and (4.15), the Fowlkes Mallows Index can derived in terms of these two measures, i.e. Precision and Recall, as given below -

$$\begin{aligned} \texttt{FM} &= \frac{\text{TP}}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN})}} \\ &= \sqrt{\frac{\text{TP}}{\text{TP} + \text{FP}} \cdot \frac{\text{TP}}{\text{TP} + \text{FN}}} \\ &= \sqrt{P \cdot R} \end{aligned} \tag{4.16}$$

To recap, in this section (Section 4.2) I have discussed three evaluation metrics. The Normalized Mutual Information evaluates the clustering in terms of the intersection of the clusters and the classes. It also stands as a harmonic mean of homogeneity and completeness measure and therefore reflects a balance of both measures. Adjusted Rand Index score indicates the accuracy of cluster assignments and Fowlkes Mallows Index indicates the geometric mean of precision and recall of the cluster assignments, and therefore reflects a balance of both measures. These three metrics are used for evaluating the clustering techniques.

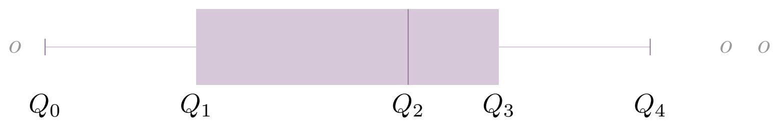
4.3 Results of Other Methods

As we discussed in the beginning, the experiments are first set up such that the documents are represented in the form of a vector. Then, the document vectors are clustered into as many clusters as there are classes in the dataset, i.e. 5, using K-Means clustering technique. However, before representing the documents as a vector, the documents require preparation and some pre-processing. In my experiments, the documents are first case-folded. Then, the documents are tokenized such that only the alphanumeric whole words of more than 2 characters are accepted as tokens. Finally, out of all the tokens that were extracted, the stopwords were removed. These preparation and pre-processing steps are the same for all the methods that will be discussed in the following sections unless specified otherwise. After documents have been tokenized, they are then converted into vector forms using each technique[4]. The vectors are then clustered using K-Means clustering algorithm (discussed in Section 3.3.2). The cluster centers are initialized using the initialization technique of K-Means++[5] and the clustering with minimum RSS[6] is selected out of 10 clusterings, each initialized with different cluster centers. The results of the clustering are then evaluated based on the evaluation metrics discussed in the previous section (Section 4.2).

For a comparative analysis, the experiments were conducted on a wide range of values for the parameters for each of the techniques discussed in Chapters 2 and 3. The methodology discussed in Chapter 3, in particular, can be applied with a lot of variations depending on the word embeddings used, the algorithm used to cluster the word vectors and the weighting function used. In this section, I will state the exact value of parameters configured for each method and discuss the performance of each technique in comparison with the rest, particularly, in comparison with the methodology discussed in Chapter 3.

Boxplots

In the tasks where labeled data is not available for the development of a problem solution, the selection of parameters for an unsupervised method can be a challenge. Therefore, a method that can consistently perform well, regardless of the parameters will be preferred. For this purpose, it is useful to analyze the performance score distribution over the experiments. In this thesis, I will use the box and whisker plots, more commonly known as *boxplots*, for such analysis since the boxplots are used to depict the distribution of data. Therefore, before we proceed to a discussion of experiments, let's briefly discuss the important attributes of a boxplot. For the discussion, a horizontal boxplot is given below -



[4]For implementation details, please refer to Implementation Details in Appendix (Appendix D)
[5]The initialization of cluster centers K-Means++ is discussed in Section 3.3.2
[6]RSS refers to "residual sum of squares". This has been discussed in Section 3.3.2

In a boxplot, a box is drawn from the 25th percentile (Q_1 or First quartile) to 75th percentile (Q_3 or Third quartile) of the distribution. In the horizontal boxplot above, the vertical line within the span of the box indicates the median, i.e. 50th percentile (Q_2 or Second quartile). The horizontal lines extending out from the box are known as the whiskers. The end of the whiskers is marked by small vertical lines or tabs. These tabs indicate the minimum and maximum values of the distribution excluding any outliers, i.e. 0th percentile (Q_0) and 100th percentile (Q_4 or Fourth quartile) respectively. All the instances that have a lower value than Q_0 or higher value than Q_4 are considered outliers. The outliers are shown as small circles outside the span of the box and whiskers. Now that we've discussed the boxplots, let's begin the discussion of the experiments and their results.

4.3.1 Document-Term Vector Space Model

In the Document-Term Association Vector representation of a document, each axis represents a term in the vocabulary. Therefore, before representing the documents as Document-Term Association Vectors, we need to extract the terms from the documents. Therefore, the document needs to be prepared and split into tokens. For the pre-processing, the experiments were performed both, with and without stemming[7]. For stemming Porter's stemming algorithm was used. The terms obtained after pre-processing the dataset can be used to represent the documents.

However, as we've discussed before, all the terms are not equally relevant. Therefore, it is preferred to select the terms that make the most effective features for representing the document. Document frequency is a good indicator of the relevance of the term in the corpus. The terms that are too frequent in the documents, e.g. stopwords, don't contribute to distinguishing the documents. Similarly, the terms that are too rare and have appeared, let's say, no more than once in the entire dataset, are also not contributing to the representation of most of the documents. So, in the experiments, the terms are selected on the basis of their document frequency. The experiments were conducted with all the terms as well as with strict and lenient document frequency ranges. In the strict document frequency range, the terms that appear in at least 5% of the documents and no more than 85% of the documents are retained, and the rest are removed. Similarly, in the lenient document frequency range, the terms that appear in at least 1% of the documents and no more than 95% of the documents are retained, the rest are removed. The restriction on document frequency also reduces the dimensionality and sparsity of the Document-Term Association Vectors which improves the quality of document vectors and subsequently, the performance of clustering.

Table 4.2 lists all the parameters and their various values that were used to create Document-Term Association Vectors during the DT-VSM experiments. After obtaining the terms, the score was calculated for each term in the document according to the *Document-Term association* parameter of the experiment. The document vector was obtained by representing each term as an axis and its corresponding score, the component along the axis. Finally, the document vectors thus obtained were length normalized.

Figure 4.3 shows the Normalized Mutual Information score distribution of the experiments in

[7]reduces the words to their root forms

Parameter	Description	Values
Document-term association	The association function for Document-Term Association Vector.	Binary, TF-iDF (Smoothened IDF)
Vector Size	Numerical value of the upper limit on the size of resulting vector where the terms are selected in the decreasing order of their frequency.	No upper limit, 500, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000
Document Frequency Range	The range of document frequency for terms.	None, Strict: 5% - 80% , Relaxed: 1% - 95%
Stemming	Stemming terms using porter stemmer.	True, False

Table 4.2: Various parameters and their values used for generating Document-Term Association Vectors of documents.

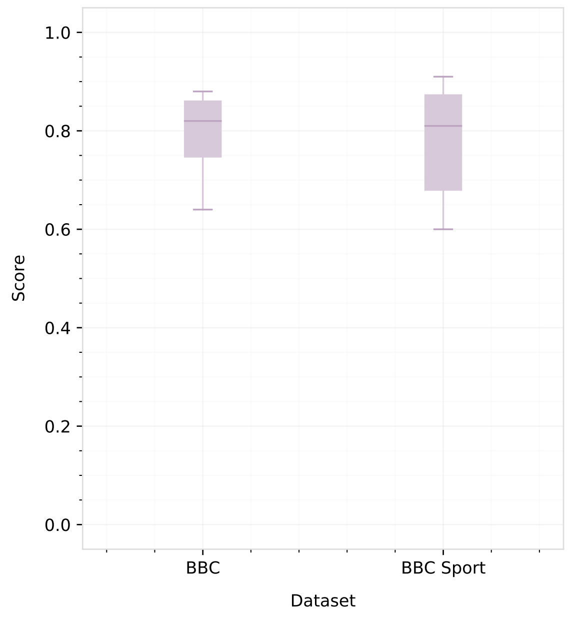


Figure 4.3: Boxplots depicting the distribution of Normalized Mutual Information scores over DT-VSM experiments on each dataset.

both datasets and Table 4.9 lists the exact Normalized Mutual Information score values corresponding to each quartiles of the boxplot. Please note that since I will mostly discuss the performance with respect to Normalized Mutual Information, only the boxplots for Normalized Mutual Information are shown in this chapter. However, the boxplots of all the performance measures for all the methods are available for reference in Appendix (Appendix A). In the Figure 4.3, the third quartile (Q_3) for Normalized Mutual Information is more than 85% for both datasets (Table 4.9). This indicates that more than 25% of all the experiments performed with a Normalized Mutual Information score of at least 85%. Moreover, Considering that the highest Normalized Mutual Information score any method achieved on BBC and BBC Sport dataset is 89% and 92% respectively, Q_3 value of 85% belonging to the DT-VSM experiments is very close to the best performance scores (Table 4.13). It is important to note here that the high values of performance indicate better clustering which in turn indicates better vector representation of the documents. Therefore, the consistency of performance for one-fourth of all experiments indicates that more than 25% of all the experiments generated document representations that were able to represent the similarity and distinction of the data points on both topical levels of news data - general and specific. Moreover, the second quartile (Q_2) being greater than 81% for both datasets indicates that half of the experiments generated satisfactory representations for both topical levels. This means that even with an uninformed selection of the parameters, half of the time the performance with respect to Normalized Mutual Information score is likely to be more than 80%.

4.3.2 Latent Semantic Analysis

As we've discussed in Section 2.2.1, in order to generate LSA vectors for the documents, first we need a term-document association data. For the term-document association data, the TF-iDF association was used. As discussed in Section 2.2.1, each column of the term-document association matrix is a document Document-Term Association Vector. Therefore, the process of building a term-document association matrix is the process of obtaining Document-Term Association Vector for each document. So, for the TF-iDF Document-Term Association Vector, the experiments were conducted over the same parameters values as DT-VSM experiments in the section before (Table 4.2), except for *Document-Term association* which was fixed to be TF-iDF for this method. Just like for DT-VSM experiments, the TF-iDF Document-Term Association Vector for LSA experiments were finally length normalized. The TF-iDF term by document matrix is then decomposed using Truncated Singular-Value Decomposition to give term-topic and document-topic matrices (Section 2.2.1). The document-topic matrix obtained after decomposition provides the representation of each document in a space of lower dimension. Table 4.3 lists all the parameters that were used in experiments conducted with the document vectors generated using Latent Semantic Analysis. The LSA Document-Topic Association Vectors obtained after the above process are then clustered and evaluated.

Figure 4.4 shows the distribution of Normalized Mutual Information score over all the experiments with LSA Document-Topic Association Vectors. As we can see from the boxplot, the NMI score corresponding to each quartile of BBC Sport dataset is higher that of BBC dataset. This indicates that the performance of LSA is better in the BBC Sport dataset than in BBC dataset.

Parameter	**Description**	**Values**
Dimension of Topical Space	The number of topical axes to reduce the original document vector to.	3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 30, 50, 70, 90, 110, 130, 150, 200, 300, 400, 500
Vector Size	Numerical value of the upper limit on the size of the resulting vector where the terms are selected in the decreasing order of their frequency.	No upper limit, 500, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000
Document Frequency Range	The range of document frequency for terms.	None, Strict: 5% - 80% , Relaxed: 1% - 95%
Stemming	Stemming terms using porter stemmer.	True, False
Document-term association	The association function for Document-Term Association Vector.	TF-iDF (Smoothened IDF)
Length Normalization	Whether the document vectors are length normalized or not	True

Table 4.3: Various parameters and their values used for Latent Semantic Analysis representation of documents

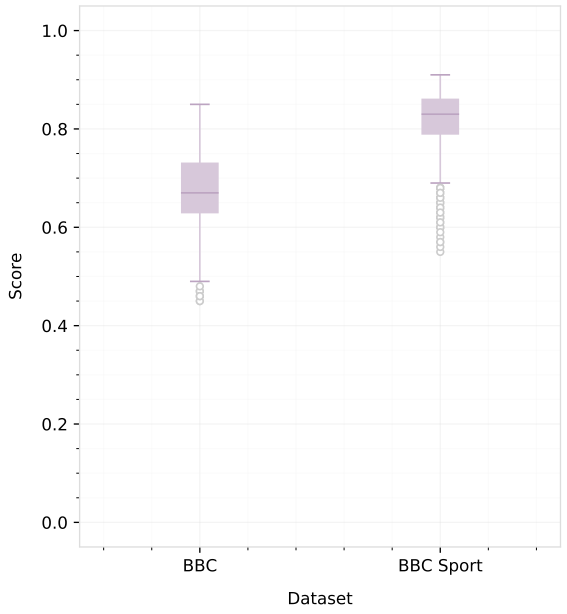


Figure 4.4: Boxplots depicting the distribution of Normalized Mutual Information scores over the LSA experiments on each dataset.

Furthermore, $Q_1 = 79\%$ of LSA experiments for the BBC Sport dataset indicates that nearly 75% of all the LSA experiments on BBC Sport dataset performed with NMI score of 79% or more (Table 4.9b). It is also worth noting from the figure that the median NMI score for BBC Sport dataset, i.e. 83%, is much higher than that of BBC dataset, i.e. 67%. When comparing with Document-Term Association Vectors, LSA is outperformed at every quartile in the BBC dataset (Table 4.9a). In terms of the best performance, or the maximum score, Document-Term Association Vectors outperform LSA vectors by a margin of 3% on the BBC dataset. However, on the BBC Sport dataset both the methods stand shoulder to shoulder with NMI score of 91% (Table 4.9).

4.3.3 Latent Dirichlet Allocation

As discussed in Section 2.2.2, in order to represent documents, Latent Dirichlet Allocation initializes two distributions -

1. Distribution of topics in documents, $\theta_d \sim \text{Dirichlet}(\alpha)$, and,
2. Distribution of terms in topics, $\beta_k \sim \text{Dirichlet}(\eta)$,

where, α and η are Dirichlet priors from which the two distributions are drawn (Hoffman et al., 2010). For simplicity, I've used symmetric priors such that

$$\alpha = \eta = \frac{1}{\text{K}}$$

where K refers to the number of topics a documents is to be represented in. Based on these distributions and the observed occurrence of terms in the documents, the conditional probability of topics given the terms in the documents is computed. In the experiments, the occurrence of the terms in the documents is observed as a binary variable. It means that the observed association between the documents and the term records only the presence of the term in the document. The remaining parameters that were used in Latent Dirichlet Allocation experiments are listed in Table 4.4.

Parameter	**Description**	**Values**
K	The number of topics axes to reduce the original document vector to.	3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 30, 50, 70, 90, 110, 130, 150, 200, 300, 400, 500
Vector Size	Numerical value of the upper limit on the size of the resulting vector where the terms are selected in the decreasing order of their frequency.	No upper limit, 500, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000
Document Frequency Range	The range of document frequency for terms.	None, Strict: 5% - 80% , Relaxed: 1% - 95%

Table 4.4: Various parameters and their values used for Latent Dirichlet Allocation representation of documents

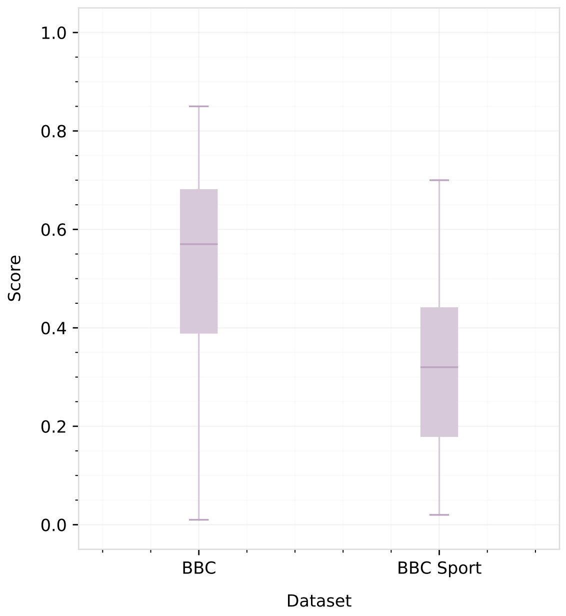


Figure 4.5: Boxplots depicting the distribution of Normalized Mutual Information scores over the LDA experiments on each dataset.

Figure 4.5 depicts the distribution of Normalized Mutual Information scores over the experiments using LDA Document-Topic Association Vectors. First of all, quite a contrast from LSA, which performs better on the BBC Sport dataset than on the BBC dataset, we can see that LDA performs better on the BBC dataset in comparison to the BBC Sport dataset. From Table 4.9, we can quickly observe that the performance of LDA varies widely for both datasets. On the BBC dataset, it goes as low as 1% and as high as 85%. Similarly, on the BBC Sport dataset, the performance goes as low 2% and as high as 70%. This range is quite large, especially in comparison to LSA and DT-VSM. In comparison to the minimum score of 1% in the case of LDA, the lowest NMI score on either dataset for LSA and DT-VSM is 48% and 60% respectively. Overall, such variability and range of scores are undesirable for unsupervised methods, because there is usually no way of selecting the parameters that will perform the best. As the concluding remark, it should be noted that LDA is outperformed by both LSA and DT-VSM, on almost every quartile, on both of the datasets.

4.3.4 Unweighted Average of Word Vectors

In order to represent the documents as a simple average of the semantic word vectors, we first need word vectors for all the words in a document. The document is then represented by an average of these word vectors as described in Section 2.3.2. For this method, pre-trained embeddings listed in Table 2.1 were used to represent the documents. In this chapter, the experiments using an unweighted average of word vectors are collectively referred to as `WE_AVG` in the plots and tables.

First of all, let's note here that the boxplots of the experiments, in this case, are not varying over a large range of scores, regardless of the pre-trained word embedding. The range of NMI scores is

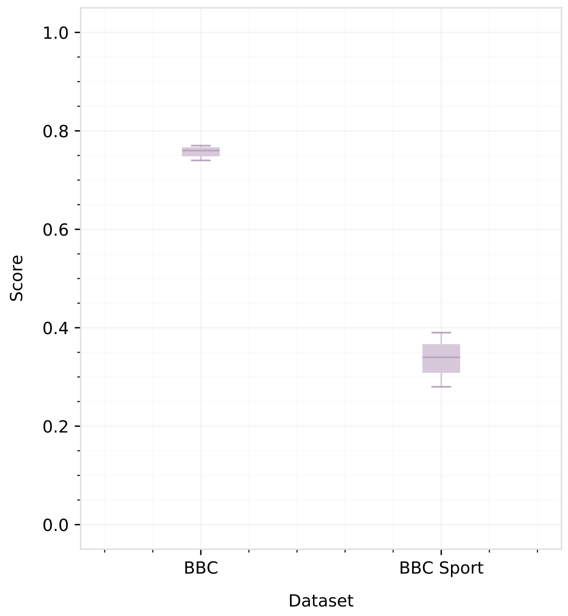


Figure 4.6: Boxplots depicting the distribution of Normalized Mutual Information scores over the `WE_AVG` experiments on each dataset.

Dataset	**Pre-trained Word Embedding**	**NMI**	**ARI**	**FMI**
BBC	GloVe 300d	0.74	0.76	0.81
	Word2vec Pre-trained	0.76	0.75	0.8
	GloVe 100d	0.77	0.78	0.82
BBC Sport	GloVe 300d	0.28	0.2	0.4
	Word2vec Pre-trained	0.34	0.27	0.45
	GloVe 100d	0.39	0.28	0.47

Table 4.5: The results of document clustering using unweighted average of semantic word vectors to represent the documents

from 74% to 77% for the BBC dataset, 28% to 39% for the BBC Sport dataset. It is also important to note here that the total number of experiments, in this case, is much lower than that for the other methods discussed previously. Due to a large number of values for multiple parameters, the number of combinations of parameters was much more for the other methods. For averaging the semantic word vectors, the only varying parameter over various experiments is the pre-trained word embedding used. This amounts to a total of 3 experiments for each dataset. Since we only have 3 experiments, there isn't enough data to compute all the quartiles. With 3 experiments, we only have the minimum, maximum, and median values. Therefore, these are the only values to be observed in the boxplot Figure 4.6 as well as Table 4.9 for this method. Since we only have a total of 6 experiments, we can directly look at the scores of each experiment. The performance scores of all `WE_AVG` experiments are given in Table 4.5.

As we can see from Table 4.5 above, the average NMI score of document clustering when this method is used to represent documents is $\approx 75\%$ on the BBC dataset, which is not the worst. Interestingly, on the BBC Dataset, this representation performs better than 75% of the LDA experiments (Q_3 NMI score for LDA on the BBC dataset is 61%). But, at the same time, the average NMI score on the BBC Sport dataset drops to $\approx 34\%$, and the maximum is no greater than 39%. It is important to note here, that there is a drastic drop in performance between the two levels of topical hierarchy. This drop indicates that the averaged word vector representation for the BBC Sport dataset was not able to properly encode the intra-class similarity, nor the inter-class dissimilarity in the documents at the 2$^{\text{nd}}$ level of topical hierarchy (sports-specific news data). Here, *intra-class similarity* refers to the similarity of documents belonging to the same class/category, and *inter-class dissimilarity* refers to the dissimilarity between documents that belong to different classes/categories. And it is exactly for the purpose of testing this that the experiment was set up to use documents at two topical levels in an unsupervised fashion. In the upcoming sections, we'll see that this drastic drop in performance is consistent for all the weighted/unweighted averaging of word vectors that we've discussed in Chapter 2.

4.3.5 TF-iDF Weighted Aggregate of Word Vectors

In the previous method, we represent a document by a simple average of semantic word vectors. In this method, the document vectors are obtained as an aggregate of semantic word vectors weighted by the TF-iDF score of each word. In the experiments, the weighted components have been aggregated using one of the two functions - sum or average. Section 2.3.2 describes the process in which the weighted components have been aggregated as an average. The experiments with this method also vary with respect to vector normalization. In other words, the experiments were conducted both, with and without length normalization of document vectors. All the parameters of the experiments with this method are given in Table 4.6. For comparative plots and tables, `WE_TFIDF` collectively refers to the experiments that are using TF-iDF weighted aggregation of word vectors for representing documents.

Figure 4.7 shows the Normalized Mutual Information score distribution of `WE_TFIDF` experiments. Just like the unweighted average of word vectors, we can clearly see a drastic drop in performance

Parameter	**Description**	**Values**
Pre-trained Word embedding	Word Embedding used for obtaining semantic word vectors	GloVe 100d, GloVe 300d, Word2vec Pre-trained
Aggregate function	Function used to aggregate the weighted components along each axis of semantic word vectors	Sum, Average
Length Normalization	Whether the document vectors are length normalized or not	True, False

Table 4.6: Various parameters and their values used for representing documents using TF-iDF weighted aggregate of word vectors

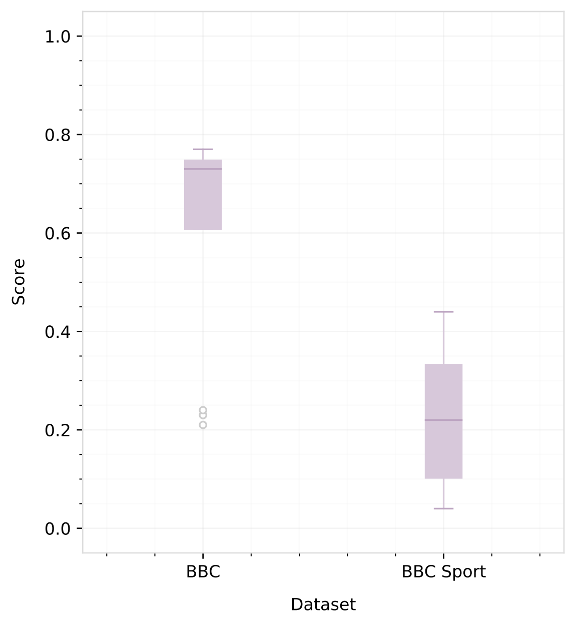


Figure 4.7: Boxplots depicting the distribution of Normalized Mutual Information scores over the `WE_TFIDF` experiments on each dataset.

from BBC to BBC Sport dataset, regardless of the quartile. Moreover, on the BBC Sport dataset, the maximum Normalized Mutual Information score achieved with this method is 44%. It can be observed that this score is significantly low in comparison to the best performing experiments using Document-Term Association Vector or LSA vectors.

4.3.6 Smooth Inverse Frequency Weighted Average of Word Vectors

For computing SIF weighted average of word vectors, we first compute weight of each word which can is calculated as -

$$f(w) = \frac{a}{(a + P(w))} \tag{4.17}$$

where,$f(w)$ is the weight of word w, $P(w)$ is the probability of word w in the entire corpus and a smoothens the inverse frequency. This smoothened inverse frequency is expected to attenuate the weights for stopwords. Therefore, in this experiment the stopwords were not removed explicitly. Moreover, in the experiments, I have experimented with different values for the smoothening term a (Table 4.7).

The probability of words can be calculated from the frequency of the words in the dataset at hand. However, we can also use frequencies computed externally to calculate the probability. SIF weighted average experiments have been conducted in both settings - with and without word frequency computed on external data. The external weights used contain words and their frequency calculated from the Wikimedia (2012). In the externally computed word frequencies, only words with a frequency of 200 or higher in the Wikimedia (2012) dataset were retained, the rest were discarded (Arora et al., 2017).

After obtaining the weighted average of semantic word vectors ad the intermediate representation of the documents, an additional step is applied. Arora et al. (2017) removed the first principal component to correct the document vector with respect to the most frequent text that is often related to syntax. In the experiments, I have varied the removal of principal components. So, the experiments are conducted with no principal component removal, 1 principal component removal, and 2 principal component removal. All these variations were used with each pre-trained embedding given in Table 2.1. Table 4.7 lists all the parameters and their corresponding values used in the experiments.

Figure 4.8 shows the boxplot of performance of all the experiments where the documents were represented using a Smooth Inverse Frequency weighted average of word vectors. Table 4.9 lists the quartile values for the boxplot where the experiments using SIF weighted average of word vectors are collectively referred to as `WE_SIF`. As we can see from Table 4.9 and figure 4.8, the maximum NMI score of BBC Sport document clustering with SIF weighted averaging is 63%, whereas it is 39% and 44% for clustering using unweighted average and TF-iDF aggregate of word vectors. Therefore, for the BBC Sport dataset, we can say that the document vectors generated by using SIF weighted averaging lead to better clustering than the ones using unweighted averaging or TF-iDF weighted aggregate of word vectors. However, based on the maximum score, the overall method of averaging

Parameter	**Description**	**Values**
a	Inverse frequency smoothening parameter	0.00001, 0.0001, 0.001, 0.01, 0.1
Removed Principal Components	The number of principal components to be removed	0, 1, 2
Pre-trained Word embedding	Word Embedding used for obtaining semantic word vectors	GloVe 100d, GloVe 300d, Word2vec Pre-trained
External Weights	Word frequencies computed on Wikimedia (2012) are used for calculating word weights	True, False

Table 4.7: Various parameters and their values used for representing document using Smooth Inverse Frequency weighted average of word vectors

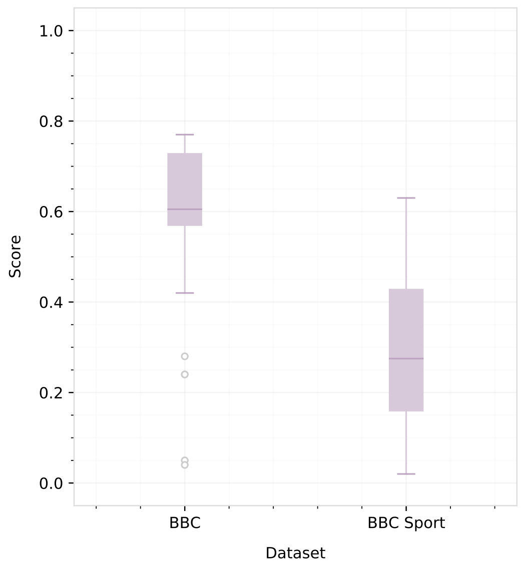


Figure 4.8: Boxplots depicting the distribution of Normalized Mutual Information scores over the `WE_SIF` experiments on each dataset.

Parameter	Description	Values
Vector Size	Size of the document vector	100, 200, 300, 400, 500
Vocab Size	Upper limit on the size of vocabulary. Top most frequent words are kept	3000, 5000, 10000
Minimum Frequency	The words with frequency lower than this parameter are ignored	2, 5
Window	The maximum distance between the current and predicted word within a sentence	5, 10
Function on Context Vectors	The function to be applied on the context vectors	Sum, Average
Epochs	Number of iterations (epochs) over the corpus	10, 30, 50, 100, 150

Table 4.8: Various parameters and their values used for representing document using Doc2vec

the word vectors, weighted or otherwise, is significantly outperformed by all the others, on each evaluation metric (Table 4.13 and figures 4.17 to 4.19).

4.3.7 Doc2vec

To compute Doc2vec vectors for the documents, the Doc2vec model was trained using the distributed memory (PV-DM) algorithm with negative sampling of 5 noise words. The noise words were selected as the negative samples based on an exponent of 0.75 of their frequency. The various Doc2vec model configurations that were experimented with, over a varying range of values for document vector size, the vocabulary, context window, function on context vectors, and the number of iterations over dataset are all listed in Table 4.8.

Before we discuss the boxplot, let's look at the best performance achieved with Doc2vec vectors (Table 4.13). The maximum NMI score for Doc2vec experiments is 84% for the BBC dataset and 89% for the BBC Sport dataset. Whereas the maximum NMI scores for DT-VSM experiments are 88% and 91% respectively (Table 4.9). This shows that the best performing Doc2vec experiment is outperformed by the best performing DT-VSM experiment. Furthermore, we can see from Table 4.13 that Doc2vec vectors are outperformed by DT-VSM by a margin of $2-5\%$ regardless of the dataset or evaluation metric.

Figure 4.9 shows the boxplot of NMI scores in Doc2vec experiments. It is important to note here that the distribution of the Normalized Mutual Information scores over all the Doc2vec experiments varies from nearly zero to more than 80%. This spread of the scores is interesting because it proves just how sensitive the performance is with respect to the selection of parameters. It also shows how bad the performance can be depending on the model configuration, especially if you have no way of evaluating which model configuration is better. Since unsupervised tasks do not have the labels against which we can evaluate the performance of a model configuration, this spread is undesirable in

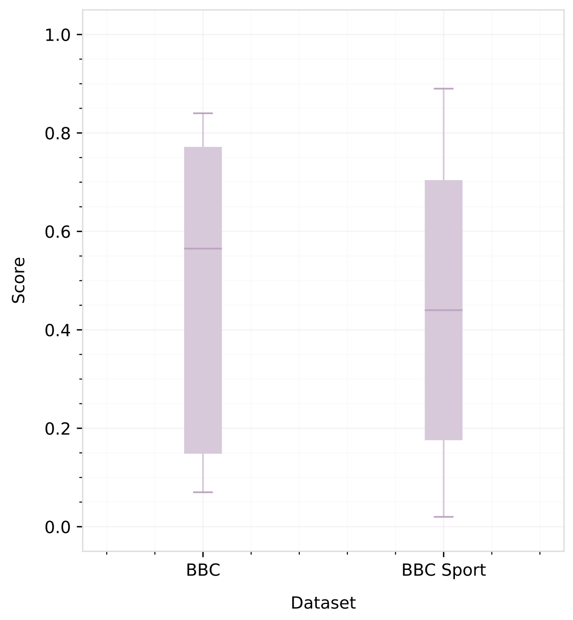


Figure 4.9: Boxplots depicting the distribution of Normalized Mutual Information scores over the Doc2vec experiments on each dataset.

a model for unsupervised analysis. So, while the best performing configuration of Doc2vec according to Table 4.13 is very close to the highest performance overall, the spread makes it less preferable than DT-VSM. While I say that, it should also be considered that the spread of performance values for Doc2vec experiments could also be related to more combinations of parameters of Doc2vec than DT-VSM. Therefore, to be fair, a more informed Doc2vec parameter selection could reduce the spread significantly and increase the minimum performance values as well.

4.4 Results of WcDe

The methods so far have been compared against each other. This section, however, focuses on all comparisons related to the WcDe methodology. Before using WcDe methodology to generate document vectors, the documents are prepared by tokenizing them and case-folding as discussed in the beginning. However, for the experiments with WcDe methodology, the stopwords are not removed. I hypothesize that the stopwords will be clustered together in the word clustering step and therefore will neither significantly increase the dimensionality of document vectors nor significantly impact the performance of the clustering. Therefore, retaining the stopwords allows for an analysis of the word clusters with respect to the stopwords later on in chapter 5. The tokenized documents are used to generate Word Cluster based Document Embedding for the documents. The experiments have been conducted both with and without length normalization of the document vectors (Table 4.10).

As we've discussed in Section 3.3, there are three essential components in the Word Cluster based Document Embedding methodology -

1. The semantic word vectors or the word embedding

	Doc2vec	LDA	LSA	DT-VSM	WE_AVG	WE_SIF	WE_TFIDF	WcDe	WcDe+
Min.[a]	0.07	0.01	0.45	0.64	0.74	0.04	0.21	0.00	0.65
Q_0	0.07	0.01	0.48	0.64	0.74	0.42	0.61	0.00	0.65
Q_1	0.15	0.39	0.63	0.75	0.75	0.57	0.61	0.10	0.74
Q_2	0.57	0.57	0.67	0.82	0.76	0.61	0.73	0.28	0.80
Q_3	0.77	0.68	0.73	0.86	0.77	0.73	0.75	0.66	0.84
Q_4	0.84	0.85	0.85	0.88	0.77	0.77	0.77	0.89	0.89
Max.[b]	0.84	0.85	0.85	0.88	0.77	0.77	0.77	0.89	0.89

(a) BBC Dataset

[a]Minimum value including the outliers
[b]Maximum value including the outliers

	Doc2vec	LDA	LSA	DT-VSM	WE_AVG	WE_SIF	WE_TFIDF	WcDe	WcDe+
Min.[a]	0.02	0.02	0.55	0.60	0.28	0.02	0.04	0.01	0.65
Q_0	0.02	0.02	0.69	0.60	0.28	0.02	0.04	0.01	0.65
Q_1	0.18	0.18	0.79	0.68	0.31	0.16	0.10	0.06	0.74
Q_2	0.44	0.32	0.83	0.81	0.34	0.28	0.22	0.18	0.81
Q_3	0.70	0.44	0.86	0.87	0.37	0.43	0.33	0.53	0.85
Q_4	0.89	0.70	0.91	0.91	0.39	0.63	0.44	0.92	0.90
Max.[b]	0.89	0.70	0.91	0.91	0.39	0.63	0.44	0.92	0.90

(b) BBC Sport Dataset

[a]Minimum value including the outliers
[b]Maximum value including the outliers

Table 4.9: Normalized Mutual Information score values for the boxplots of each method. The color-coding of the cells containing the score is distributed such that the minimum score is darkest red, and the maximum score is the darkest green. The median of the color distribution (white) has been fixed as 0.75.

2. The clustering method used for clustering the semantic word vectors
3. The weighting scheme that attributes a numerical score to the relationship between the word cluster and the document

The first two define the WcDe vector space and the last one defines the component of each document vector along an axis in this vector space. There are many possible options for each of these components which lead to multiple variations based on the same fundamental methodology (discussed in Section 3.3). Next, I'll discuss these options one component at a time and specify the experiment parameters (Table 4.10).

Parameter	Description	Values
Word Vectors	Vectors for words	GloVe 100d, GloVe 300d, Word2vec Pre-trained, Trained Word2vec SGNS (20, 50, 100 and 150 epochs)
Clustering Method	Method used to cluster word vectors	K-Means, Agglomerative Hierarchical Clustering
Association Functions	The function used to associate a cluster score for the document with the cluster	CF-iDF, TF-iDF Sum
Length Normalization	Whether the document vectors are length normalized or not	True, False

Table 4.10: Various parameters and their values used for representing document using WcDe methodology

Semantic Word Vectors/Word Embedding In Section 3.3.1, I've discussed WcDe methodology with respect to its first component, i.e. semantic word vectors. In that section, one of the things that we discussed was the use of semantic word vectors trained from scratch and pre-trained word embedding in the WcDe methodology. In my experiments, I have used both. For pre-trained word embeddings, I used the ones listed in table 2.1. In addition to the pre-trained word embeddings, I trained 300-dimensional word vectors using Skip-gram Word2vec architecture with negative sampling (SGNS) with a context window of 10 words. For negative sampling, 5 noise words were sampled for each training sample with the proportion based on an exponent of 0.75 of their frequency. Due to the sheer number of parameters to be experimented on in WcDe methodology, the Word2vec model parameters were fixed. Using the same model parameters given above, 4 Word2vec models were trained for a different number of epochs - 20, 50, 100 and 150 (Table 4.10). However, it is not without reason that only epochs were varied for the experiments.

The purpose of experimenting with pre-trained as well as the word vectors trained from scratch was to observe whether the semantic word vectors trained for specific data offer an advantage over the pre-trained ones in WcDe methodology. Training for a higher number of epochs is likely to overfit the word vectors for the dataset at hand. In supervised tasks, overfitting is undesirable because the prediction model learns weights specific to the data. This leads to poor performance on unseen data. However, in an unsupervised task like document clustering, no weights are learnt in a supervised way and there is no unseen data that we will evaluate the methodology on. In an unsupervised task, like document clustering, it is probably better to have word representations that are less generalized and more specific to the dataset. Training the word vectors on a selective dataset might provide the advantage of having word vectors computed through the contexts that are more pronounced and consistent through the dataset at hand. Therefore, keeping the above considerations in mind, the word vector training was varied with respect to epochs, specifically. For future work, the Word2vec model can be tuned on parameters that suit the dataset and task better and even be trained with additional datasets for better results.

Clustering Algorithm for clustering word vectors For the scope of this thesis, the clustering algorithm used to cluster semantic word vectors in WcDe experiments have been limited to - K-Means and Agglomerative Hierarchical Clustering. Using Agglomerative Hierarchical Clustering, the word clusters were created with various values of distance threshold as well as merge criteria for merging the word clusters. Using K-Means, the semantic word vector space was clustered into various pre-specified number of clusters. For K-Means, the word cluster centers were initialized using the initialization technique of K-Means++[8] and the word clustering with minimum RSS[9] is selected out of 10 clusterings, each initialized with different cluster centers. Table 4.11 lists the clustering configurations used in WcDe experiments for each of the clustering algorithms.

Clustering Method	**Parameter**	**Values**
Agglomerative Hierarchical Clustering	Distance Threshold	4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 9.5, 10.0, 10.5, 11.0, 11.5, 12.0
	Merge Criteria	Average Similarity, Maximum Distance (Minimum similarity), Minimum Variance (Ward)
K-Means Clustering	Number of Clusters	100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000

Table 4.11: Various clustering parameters and their values used in WcDe experiments for representing documents

The weighting function After clustering the semantic word vectors, the WcDe document vector space is defined such that each axis represents a cluster of words. Now, to represent each document in this vector space, a weighting function is applied to calculate the score of the word cluster in the document. This score serves as the component along the axis that represents the word cluster. I have discussed a few weighting functions in Section 3.3.3. In my experiments, I have used CF-iDF

[8]The initialization of cluster centers K-Means++ is discussed in Section 3.3.2
[9]RSS refers to "residual sum of squares". This has been discussed in Section 3.3.2

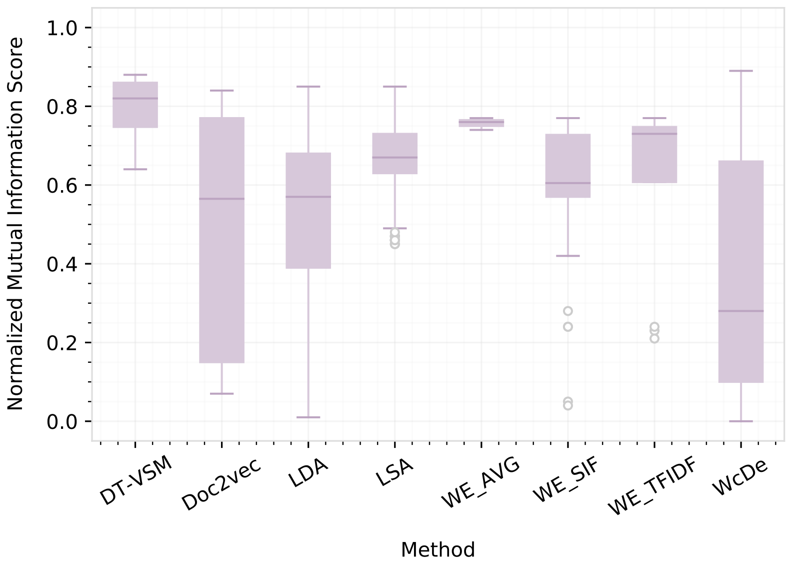
and a simple sum of TF-iDF weights to calculate the weight of the word cluster in the document (Table 4.10).

Figure 4.10 shows the boxplots of Normalized Mutual Information score of all the methods on both datasets. As we can see from the boxplot corresponding to WcDe, the performance can range from the best to worst. The selection of word vectors, clustering methods, and weight function for WcDe significantly impacts the performance. As we can see from Table 4.10, the experiments for this method have multiple options for each component. Furthermore, each of these options can be further configured. This results in a large number of experiment configurations that were tried with WcDe methodology. Some options are less optimal than others. In WcDe we can analyze each component, look at the document vectors to further optimize the document representation for our purpose. However, it is important to note here that in Doc2vec vectors, because of the abstraction of features, we can not analyze or improve them in a similar fashion. The only manner of improvement of document representation with Doc2vec can either come from the dataset or the model parameters. The beauty of using WcDe is in exactly this aspect of it - while it makes use of the contextual information encoded in semantic word vectors, the document features are not abstract. Both the document features and the components of the WcDe methodology can be inspected for further areas of improvement. In the remainder of this thesis, I'll do exactly that. In the next section, I'll inspect each component, and in the next chapter, I'll inspect the document vectors. As we'll see soon, by analyzing the components and their various configurations, we'll gather consistently well-performing configurations for each component and we'll be able to significantly limit the performance range.

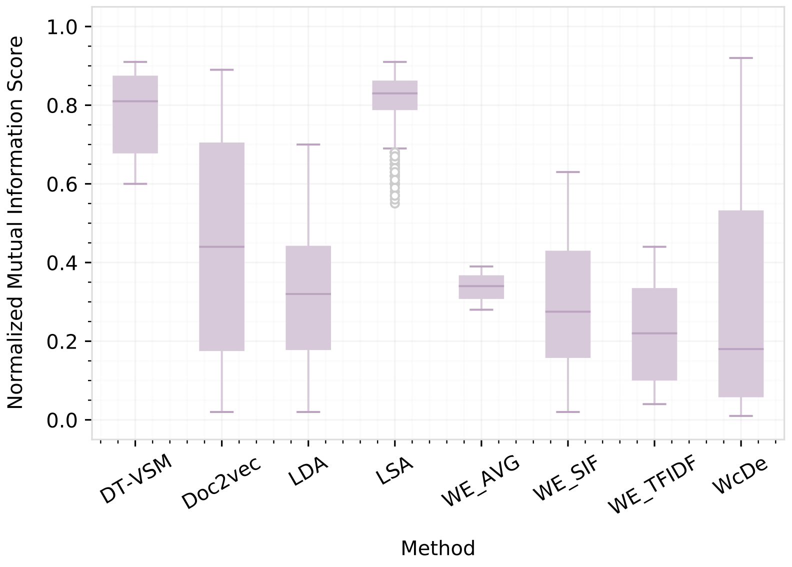
4.4.1 WcDe Variations

In my experiments with WcDe, the most apparent difference in performance was observed in the case of various weight functions and length normalization. Therefore, even though these contribute to the last step of the methodology, I will begin this analysis and discussion pf components with weight functions and length normalization. Figure 4.11 (page 77) shows the distribution of Normalized Mutual Information score for a combination of weight functions and length normalization. Quite interestingly, it can be observed that except for a few outliers, the length normalized CF-iDF document vector clearly outperforms the WcDe vectors with the rest of the weight function and length normalization configurations, on all quartiles regardless of the dataset. This observation will now be used to further analyze and find such patterns for other components of the WcDe methodology within the experiments. I will analyze the performance of the rest of the components for the subset of all WcDe experiments that used CF-iDF for weighting the features and applied length normalization to the document vector.

The next component of WcDe that we will analyze is the clustering method used to cluster the word vectors. The boxplots in Figure 4.12a (page 78) depict the distribution of performance of all the WcDe experiments for various clustering methods. But as we can see from the boxplots, the results vary over a very large range. Yet, with a quick look at Figure 4.12a we can see that for the BBC dataset, the experiments using K-Means clustering and AHC with minimum variance merge

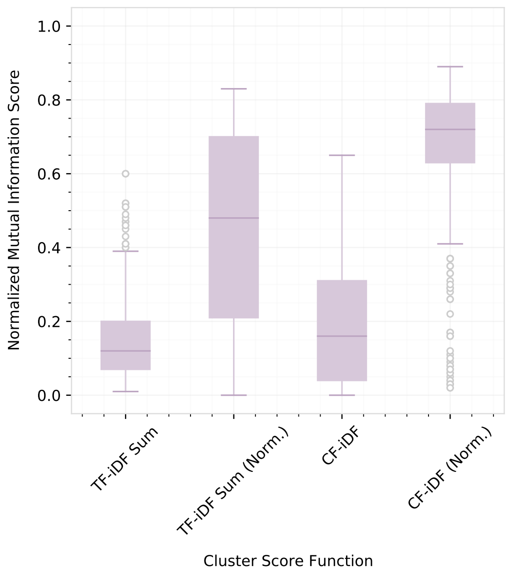


(a) BBC dataset

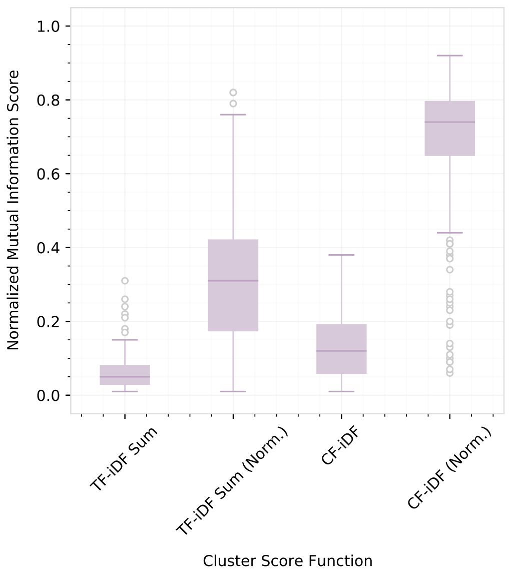


(b) BBC Sport dataset

Figure 4.10: Boxplots of Normalized Mutual Information of all methods on both datasets



(a) on BBC Dataset

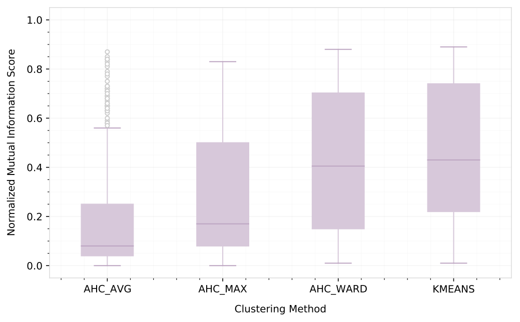


(b) on BBC Sport Dataset

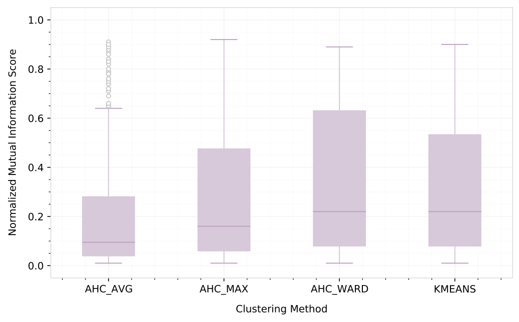
Figure 4.11: Boxplots depicting the distribution of Normalized Mutual Information score with respect to a combination of weighting functions and length normalization for the WcDe experiments

criterion for word clustering have relatively higher values for the median. Moreover, the Fourth quartile Q_4 value for both is among the highest and the size of the upper whisker (Q_3 to Q_4) is relatively small. This indicates that the experiments indicated by the span of the upper whisker had less variance and high performance on average. For the BBC Sport dataset, however, the medians are all really low and the maximum values are mostly shoulder to shoulder. Overall, there is no obvious pattern that can be observed from Figure 4.12a for the BBC Sport dataset.

For a much clearer pattern, let's focus on the experiments that use normalized CF-iDF for assigning a cluster weight (Figure 4.12b). In other words, instead of all the WcDe experiments, let's investigate the clustering algorithms on a subset of WcDe experiments. Based on the previous analysis of weight function and length normalization, let's investigate the clustering algorithms on the subset of WcDe experiments that are using normalized CF-iDF weights. As we can clearly see from comparing the boxplots in Figures 4.12a and 4.12b, the experiments using normalized CF-iDF not only confirm the impression we can observe from boxplots for unfiltered data in Figure 4.12a, but also add to the confidence of the observation by highlighting just how much better the results are when the word vectors are clustered with K-Means clustering or AHC with minimum variance merge criterion. In both datasets, both K-Means and Agglomerative Hierarchical Clustering with minimum variance merge criterion, have higher values for nearly all quartiles. More importantly, there is a stark drop in the variance of the performance. Moreover, a closer look at the clustering parameters of all configurations (Figure 4.13), shows a consistently high performance regardless of the clustering parameter in case of both K-Means clustering and AHC with minimum variance merge criterion. It brings us closer to finding combinations of options for the components of WcDe

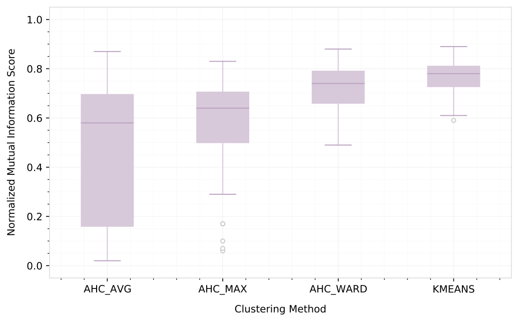


BBC dataset

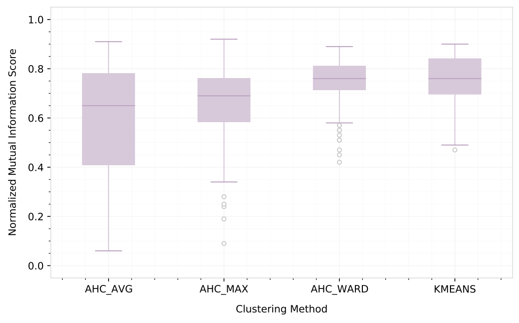


BBC Sport dataset

(a) All experiments



BBC dataset



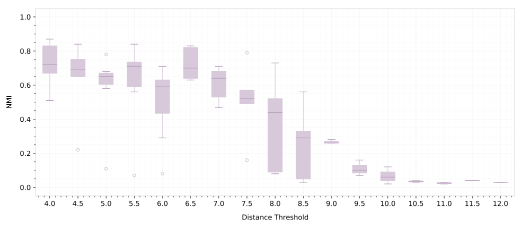
BBC Sport dataset

(b) Experiments using normalized CF-iDF

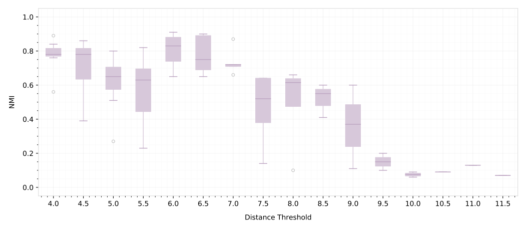
Figure 4.12: Boxplots depicting the distribution of Normalized Mutual Information score with respect to various clustering methods for the WcDe experiments

methodology that perform consistently well, with the least variance of performance.

Having analysed clustering methods and weight functions, the only component that remains is the word vectors that are used in the WcDe methodology. Figures 4.14 and 4.15 (Pages 80 and 81, respectively) show the distribution of NMI for both - the experiments using K-Means clustering and Agglomerative Hierarchical Clustering with minimum variance merge criterion in separate plots. Since the actual values can be difficult to determine from the boxplot, the values corresponding to the First, Second and Third quartile as well as the minimum and maximum values including the outliers are given in Appendix in Table B.1 (Page 111) for reference. Now with respect to selecting a word embedding for WcDe, the preference would be for the one that leads to high performance scores, regardless of the dataset. Based on the maximum and median performance scores in Figure 4.14, Word2vec trained for 20 epochs clearly dominates the performance in case of BBC dataset for both clustering methods. However, based on the same quartile scores for the BBC Sport dataset, it falls to the 2nd or 3rd place in comparison to other word vectors when clustered with AHC with minimum variance merge criterion. But it can be observed from the Table B.1 that Word2vec trained for 20 epochs has the highest scores in most of the quartiles for both datasets as well as both clustering. From Figures 4.14 and 4.15 and table B.1 we can also observe that though the quartile scores for

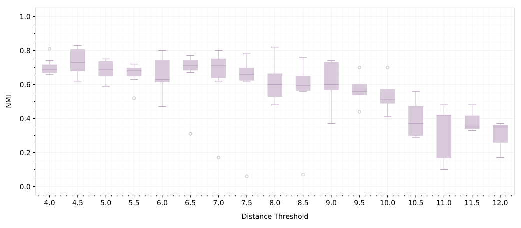


BBC dataset

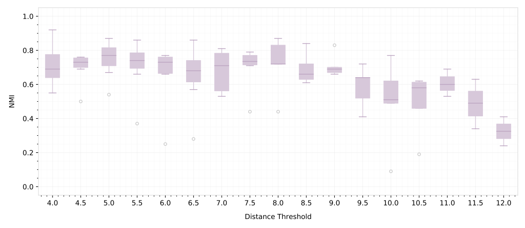


BBC Sport dataset

(a) Variation of distance threshold for Agglomerative Hierarchical Clustering with average similarity merge criterion

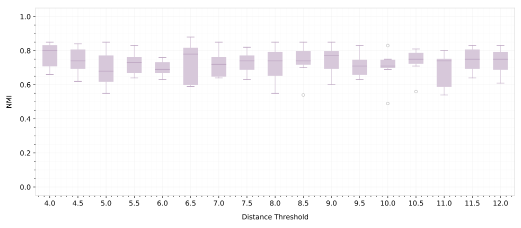


BBC dataset

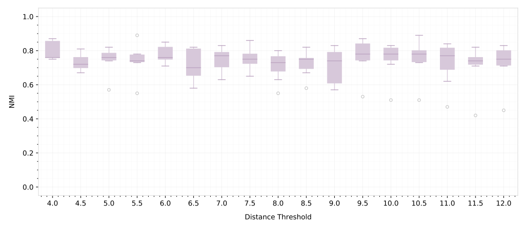


BBC Sport dataset

(b) Variation of distance threshold for Agglomerative Hierarchical Clustering with maximum distance (minimum similarity) merge criterion

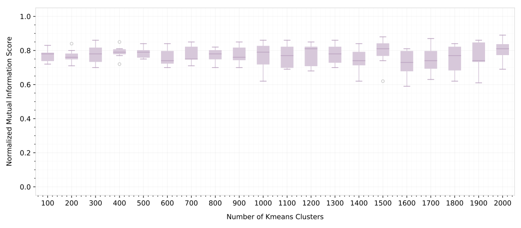


BBC dataset

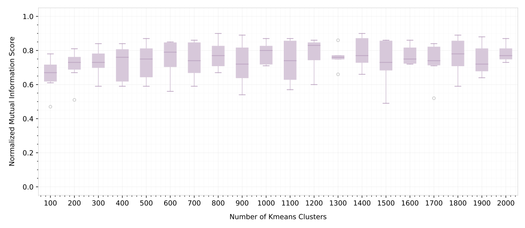


BBC Sport dataset

(c) Variation of distance threshold for Agglomerative Hierarchical Clustering with minimum variance (ward) merge criterion



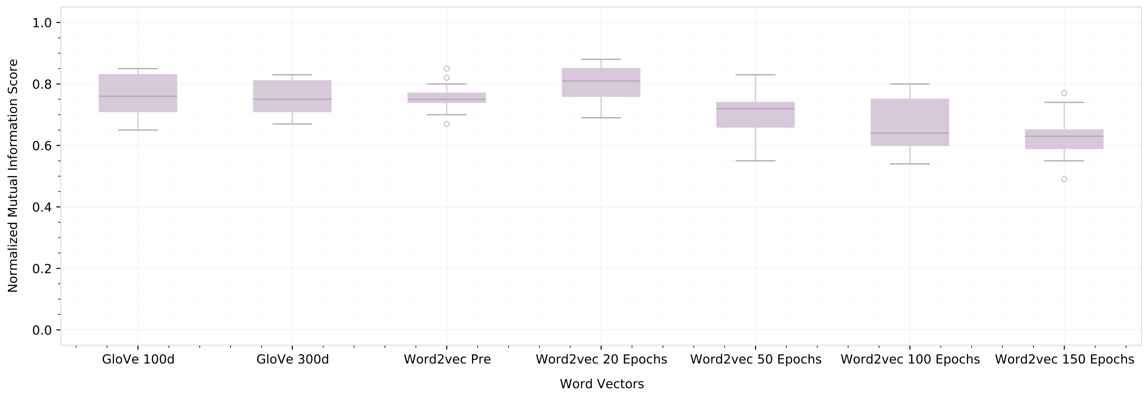
BBC dataset



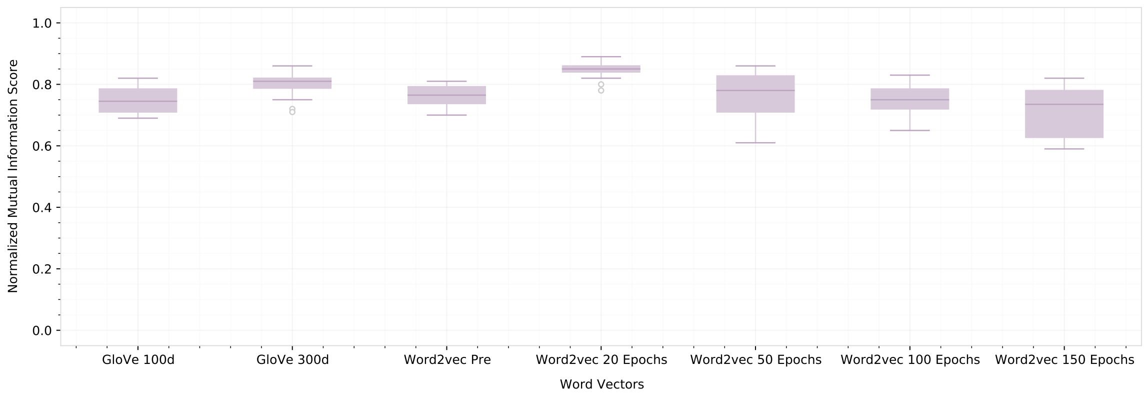
BBC Sport dataset

(d) Variation of number of clusters for K-Means clustering

Figure 4.13: Boxplots depicting the distribution of performance of WcDe experiments with normalized CF-iDF weights for variations with respect to clustering parameters.

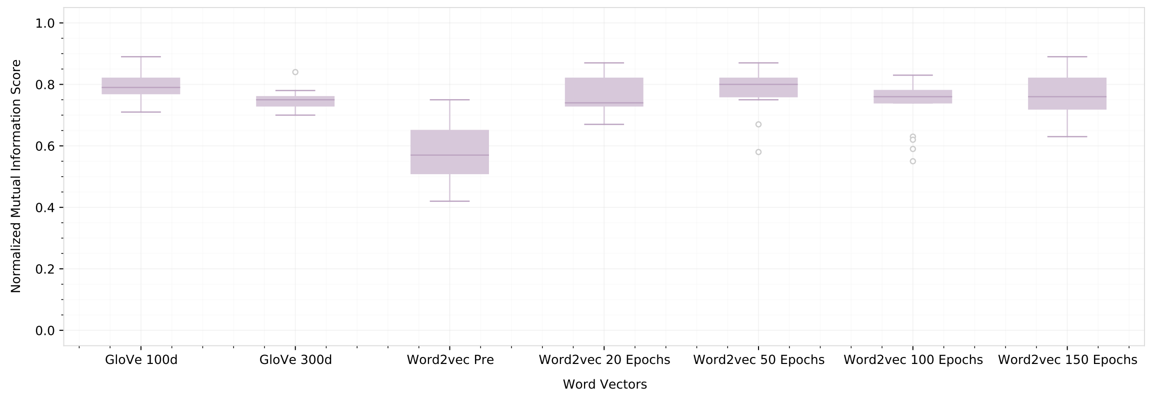


(a) Word vectors clustered using AHC with minimum variance merge criterion

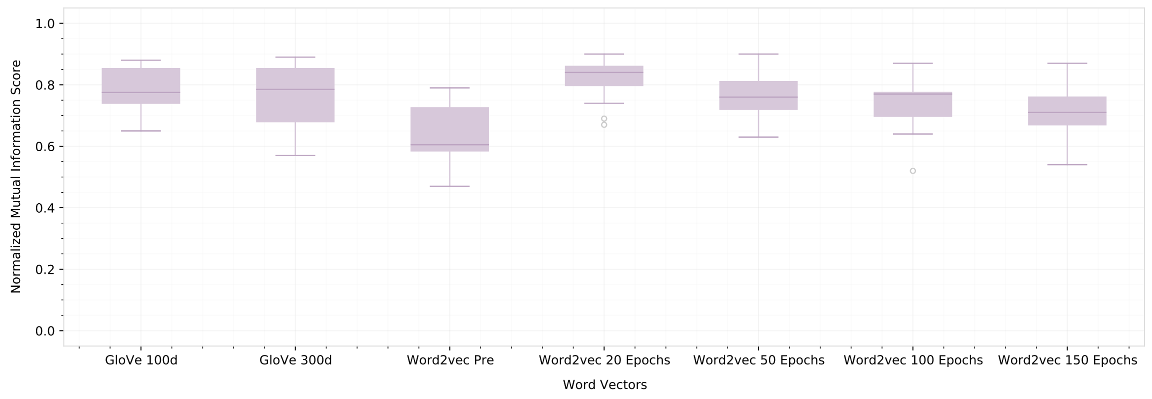


(b) Word vectors clustered using K-Means clustering

Figure 4.14: Boxplots depicting the distribution of Normalized Mutual Information score with respect to various word vectors in WcDe experiments on BBC Dataset



(a) Word vectors clustered using AHC with minimum variance merge criterion



(b) Word vectors clustered using K-Means clustering

Figure 4.15: Boxplots depicting the distribution of Normalized Mutual Information score with respect to various word vectors in WcDe experiments on BBC Sport Dataset.

	Min.	First	Second	Third	Max.
GloVe 100d	0.68	0.73	0.77	0.82	0.86
GloVe 300d	0.66	0.73	0.77	0.81	0.86
Word2vec Pre-trained	0.57	0.64	0.67	0.73	0.80
Word2vec 20 Epochs	0.70	0.78	0.81	0.85	0.89
Word2vec 50 Epochs	0.59	0.71	0.77	0.80	0.87
Word2vec 100 Epochs	0.57	0.69	0.73	0.77	0.83
Word2vec 150 Epochs	0.56	0.65	0.71	0.75	0.84

Table 4.12: Average of Minimum (including outliers), Maximum (including outliers), First, Second and Third quartile NMI scores for a subset of WcDe experiments. The values are averaged for both datasets. In this subset of WcDe experiments, the semantic word vectors are clustered with either K-Means or AHC with minimum variance merge criterion and the document vectors are created using normalized CF-iDF weights.

experiments with Pre-trained Word2vec are competitive on BBC dataset, they are the lowest among all the semantic word vectors/word embedding options for BBC Sport dataset. Moreover, based on the average performance in each quartile across both datasets and both clustering algorithms given in Table 4.12, the experiments with Word2vec trained for 20 epochs have the highest average in each quartile. The experiments with pre-trained 100-dimensional GloVe have the second-highest average for all but the Maximum, where it marginally loses to the experiments with Word2vec trained for 50 epochs. Based on the analysis so far, the Word2vec trained for 20 epochs seems to be the best choice, pre-trained 100-dimensional GloVe word embedding being the next best.

Based on the analysis of all the components so far, Figure 4.10 can be plotted again, however, this time with experiments selected on the basis of some insights drawn for each component above. I have denoted a subset of WcDe experiments as WcDe+ which use the following components -

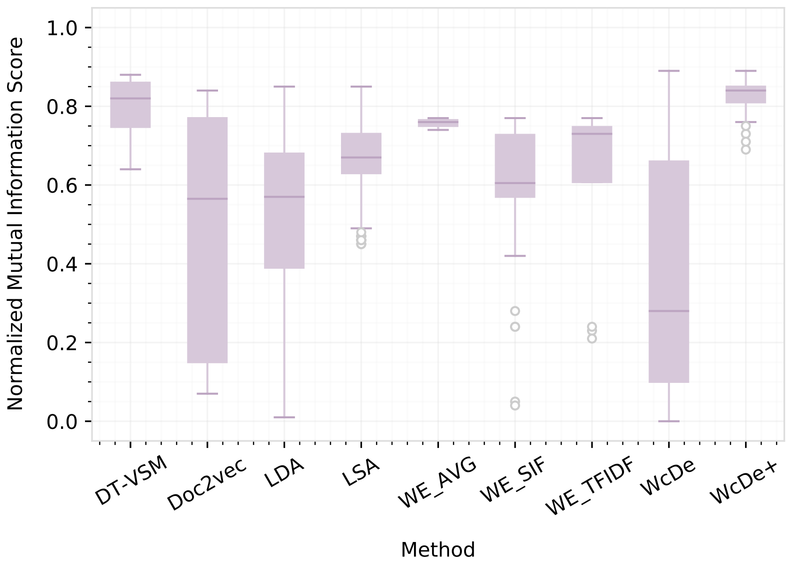
Semantic word vectors/Word Embedding Word2vec word vectors trained for 20 epochs

Clustering algorithm to cluster word vectors K-Means or Agglomerative Hierarchical Clustering with minimum variance merge criterion

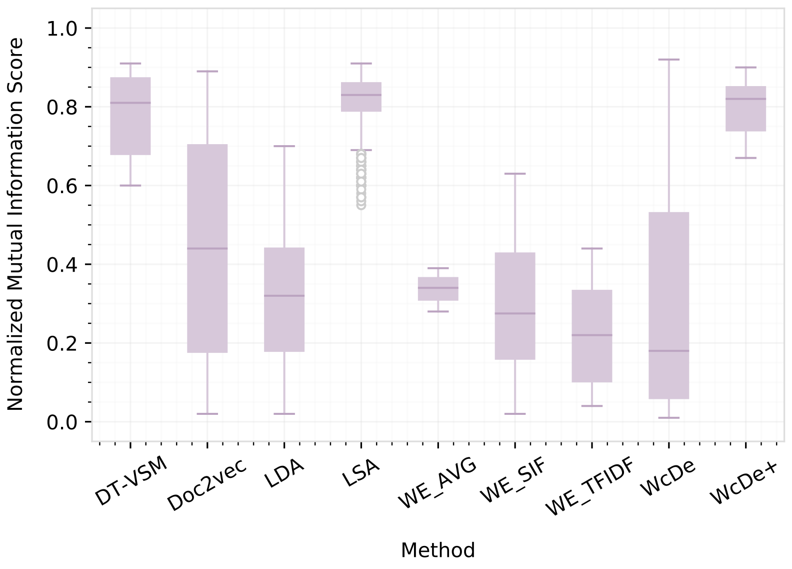
Weight Function CF-iDF

Length Normalized Yes

Figure 4.16 (Page 83) shows a contrast in the performance distribution with both - all the WcDe experiments and the WcDe+ experiments. The boxplot for WcDe+ shows distribution with high values for performance for the entire set of experiments, especially in the case of the BBC dataset. It is important to note here that limiting the experiments with some parameters, possibly excludes come configurations that had a competitive performance. Therefore, the experiments that performed the best, may not be a part of WcDe+ experiments. However, the goal of this analysis is to not select the configuration that has the maximum performance score, once. Au contraire, the goal is to find configurations that consistently perform well and show little variance with respect to parameters



(a) BBC Dataset



(b) BBC Sport Dataset

Figure 4.16: WcDe+ vs the Rest - Boxplots

like the number of word clusters, i.e. dimensionality of document vector space. And WcDe+ is able to do that for both topical levels.

4.4.2 WcDe vs the Rest

As we've discussed, the performance of many methods dropped drastically in the case of the BBC Sport Dataset. So far, we've analysed the distribution. But for a clearer comparison, let's take the best performing experiments for each method (Figures 4.17 to 4.19). It can be seen clearly in Figures 4.17 and 4.19 that there is a drastic drop in performance when using any sort of average of semantic word vectors for representing documents. However, LSA, DT-VSM, Doc2vec and WcDe have maintained the performance through both datasets. Furthermore, Figure 4.18 and table 4.13 show that even though LSA, DT-VSM, Doc2vec perform well on both the datasets, WcDe marginally outperforms all the rest in each metric on both the datasets.

Method	**NMI**	**ARI**	**FMI**
Doc2vec	0.84	0.87	0.90
LDA	0.82	0.85	0.88
LSA	0.85	0.87	0.90
DT-VSM	0.88	0.91	0.93
WE_AVG	0.77	0.78	0.82
WE_TFIDF	0.77	0.78	0.83
WE_SIF	0.77	0.78	0.82
WcDe	0.89	0.92	0.93

(a) BBC

Method	**NMI**	**ARI**	**FMI**
Doc2vec	0.89	0.89	0.92
LDA	0.71	0.73	0.79
LSA	0.91	0.92	0.94
DT-VSM	0.91	0.91	0.93
WE_AVG	0.39	0.28	0.47
WE_TFIDF	0.44	0.33	0.49
WE_SIF	0.63	0.5	0.61
WcDe	0.92	0.93	0.94

(b) BBC Sport

Table 4.13: Performance scores for each method correspond to the experiments with best performance on Normalized Mutual Information score.

In this chapter, we've compared WcDe methodology with various other unsupervised document representation techniques in a task of document clustering at two levels of topical hierarchy. Furthermore, we've drawn insight from the analysis of WcDe experiments for a consistently well-performing configuration, denoted by WcDe+. This configuration was able to significantly limit the variance of performance while maintaining high performance at both topical levels. Lastly, based on the best performance in terms of Normalized Mutual Information score, WcDe outperformed the rest of the methods on all evaluation metrics. This concludes the quantitative analysis of WcDe methodology. In the next chapter, we'll take a close look at the WcDe document vector itself to discuss how well it serves as a vector representation of documents.

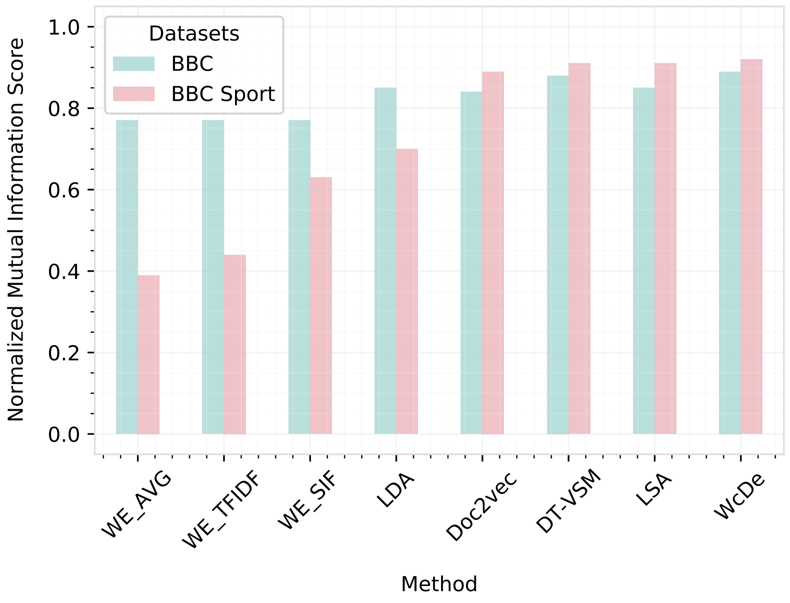
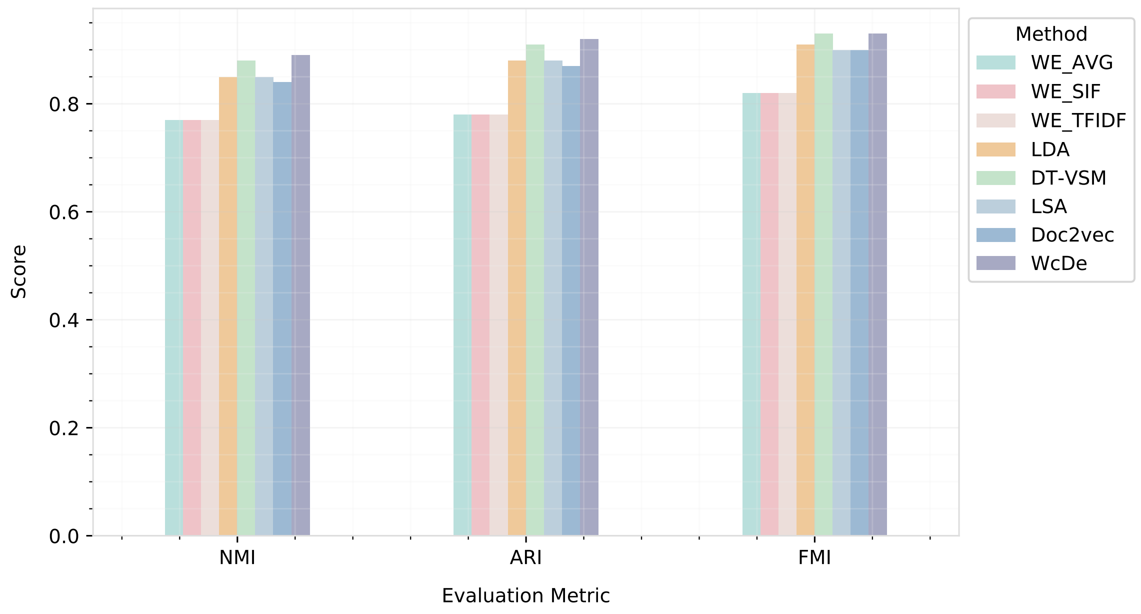
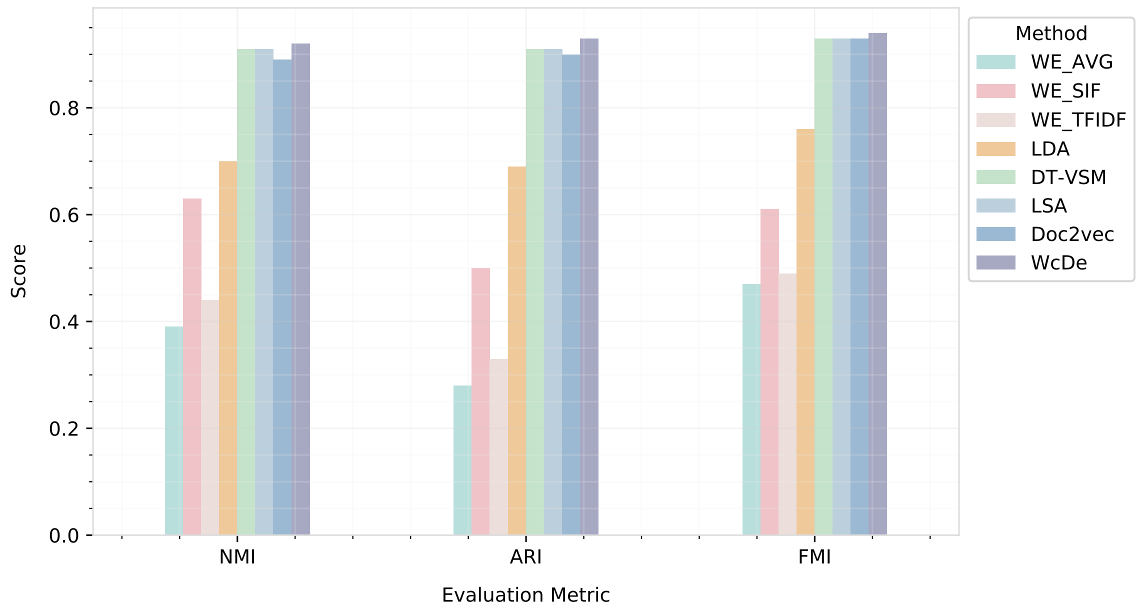


Figure 4.17: Bar graph comparing the performance of each method on both datasets side-by-side.

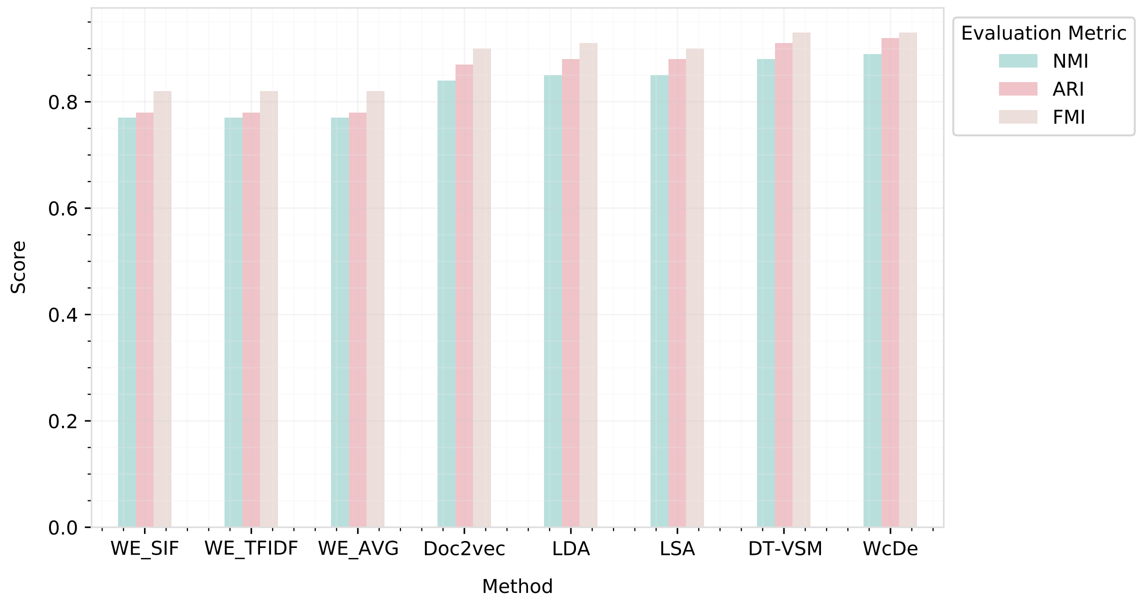


(a) BBC Dataset

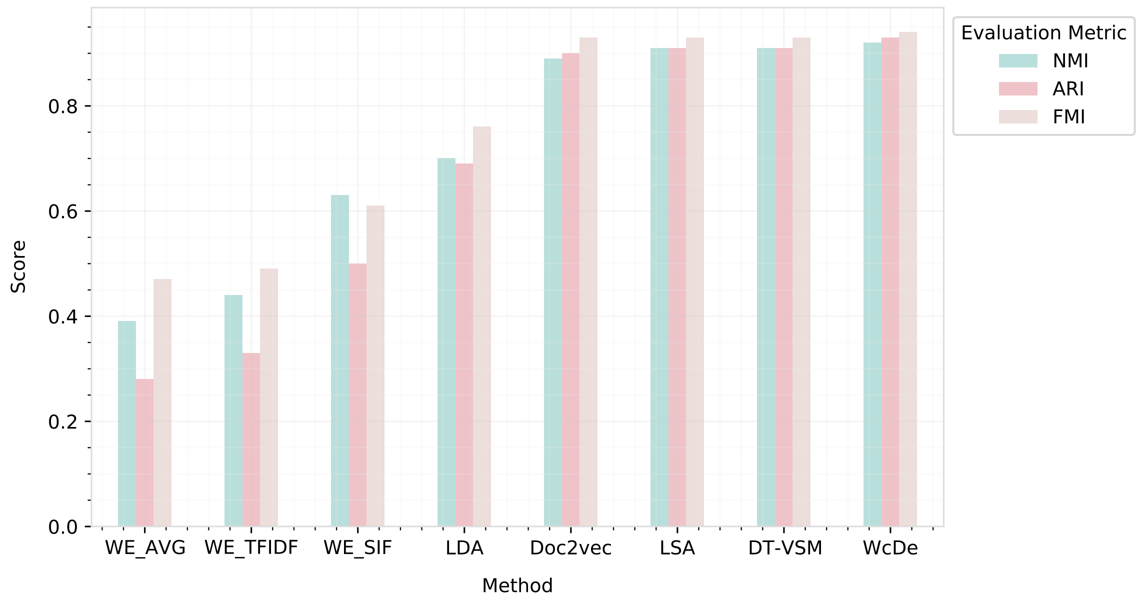


(b) BBC Sport Dataset

Figure 4.18: Bar graphs comparing the methods on each metric



(a) BBC Dataset



(b) BBC Sport Dataset

Figure 4.19: Bar graphs showing performance of each method on each evaluation metric

Chapter 5

Discussion & Qualitative Analysis

The previous chapter focuses largely on the experiments and a quantitative analysis with respect to the performance in clustering the documents. But as to whether the document vectors generated by WcDe actually capture what it was initially designed for, and if it does, to what extent - remains to be answered. And in this chapter, I hope to explore just that. In this chapter, we will take a closer look at the document vectors generated by WcDe.

5.1 Captures Similarity?

In the vector representation of data, we hope to have features that strike the right balance between capturing similarity and difference, especially for our purposes and tasks. In that respect, in the design of WcDe representation, it is assumed the documents with similar content will use similar words and therefore, have higher weights for common word clusters. Similarly, the documents with very different content will use different words and will have higher weights in different word clusters. Let's use the temperature term "hot" to speak about the value of weights corresponding to a word cluster such that a word cluster with non-zero weight in a document is hot in the document, the one with larger weight is "hotter" and the one with the largest weight is "hottest". Since the word clusters are represented by axes in the WcDe document vector space, the same applies to the discussion of axes. In other words, to speak about the value of component along an axis such that in a vector, the axes which have a non-zero components are considered hot, the one with the larger component is hotter and the one with the largest component is the hottest. So to rephrase what I said before, in WcDe representation, it is expected that the same axis will be "hotter" for the documents that belong to the same class. On the other hand for documents belonging to different classes, such pattern would be exhibited by different axes. So let's begin with the investigation for the common axes that are hotter for the documents of the same class. Since we expect the same axis to be hot in all the documents of the same class, the first line of investigation is -

- Considering all the documents of a particular class, are there any axes that are consistently hotter than the rest of the axes?

- If there are, then what are the constituents of the word clusters represented by these vector axes?

The simplest intuition for this investigation suggests that an axis that is hot (i.e. simply non-zero), for most of the documents of a class, might be strongly related to the class. However, there are two things to consider -

- The stopwords have not been removed while representing documents using WcDe. Therefore, it is very likely that the axes that are hot in nearly every WcDe document vector represent the word clusters that are made up largely of stopwords.
- It is not just the fact that a cluster is hot that counts, but also its degree. Not to mention that the hotness of a word cluster in the documents of one class does not preclude its hotness in the documents of other classes, nor its degree.

So, to begin the analysis, let's use the document vectors generated with specific WcDe configuration. Among the several available options for variations, for our analysis in this chapter, based on the analysis in Section 4.4.1, let's use Agglomerative Hierarchical Clustering with minimum variance merge criterion to obtain word clusters from pre-trained GloVe word embeddings (100 dimensional) and generate document vector by using normalized CF-iDF to compute association score for each cluster. Furthermore, since a high distance threshold results in fewer word clusters and the count of word clusters define the dimensionality of WcDe vector space, a distance threshold of 9 was used to get the word clusters. Word clustering resulted in less than 2500 clusters for both datasets. The selected distance threshold does not result in the best results and performs $80 \pm 5\%$ with respect to Normalized Mutual Information score on both datasets. The reason behind this selection was also so that we can analyze the document vector for a partly informed and partly random configuration instead of the one that we know performs the best, because this knowledge of which configuration performs best is a result of the feedback in the form of true labels. This feedback is not available in unsupervised tasks. Finally, the WcDe document vectors generated using this configuration are then used for the current analysis.

A lot of the upcoming discussion will be based on the heatmaps like the one in Figure 5.1. Therefore, before we begin, we must discuss what the heatmap represents as well as some necessary notation for the discussion. Let's take Figure 5.1, for example, which shows a heat map representation of all the document vectors belonging to `tech` class. On the x-axis, we have document vector axes which ultimately represent a word cluster. On the y-axis, we have all the documents belonging to the `tech` class, represented by an identification number that is associated with the same document throughout our discussion in this chapter. In the discussion, I'll refer to the document with id 1825 (y-axis) as `DOC-1825`. In the heatmap, the color scale is given on the right which gives the color corresponding to the component value of the document vector axis. In other words, the color indicates the weight of the word cluster in the document. Finally, in the discussion in this thesis, especially this chapter, I will repeatedly refer to the numbers that represent the axes of the WcDe vector space (x-axis of heatmap). The axes represent word clusters, and in that sense, the numbers on the x-axis of the heatmap also represent the word clusters. However, in the discussion, we have two datasets and therefore two different sets of word clusters. Therefore, in the discussion, I'll refer to the axes or the word clusters with dataset names prefixed to the number. For example, for the BBC dataset, the axis labeled 1192 and its corresponding word cluster will be referred to as `BBC-1192` and, similarly for the BBC Sport dataset, they will be referred to as `BBC-SPORT-1192`. To sum it up with an example, in Figure 5.1, the color corresponding to row labelled 1825 and column labelled 981 indicates the weight of word cluster `BBC-981` in `DOC-1825` as per the scale on the right.

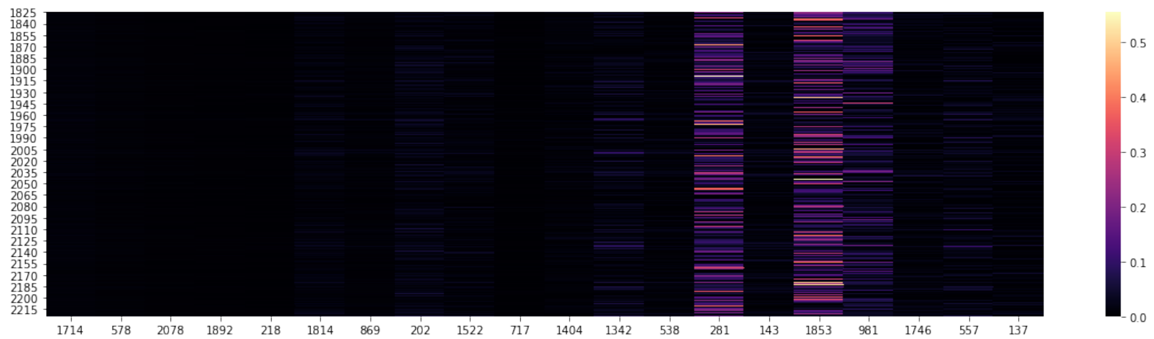


Figure 5.1: Heatmap of top 20 axes of WcDe document vector representation of all documents of class `tech`. The axes have been sorted from left to right in the decreasing order according to Count$_{>0}$.

Axis	**Count$_{>0}$**	**Sum**	**Max**
1714	400	3.84	0.02
578	400	2.49	0.02
2078	400	1.68	0.01
1892	400	1.44	0.01
218	400	1.28	0.01
1814	394	4.98	0.04
869	394	2.54	0.03
202	386	7.04	0.06
1522	385	4.47	0.04
717	377	1.35	0.01

Table 5.1: Statistics for the top 10 axes (word clusters) for `tech` class. The axes have been sorted in the descending order of the count of document vectors in which the component along the axis a non-zero value (Count$_{>0}$).

A paragraph ago, we discussed the intuition that the axis (of WcDe vector space) with non-zero components for most documents of a class, might be strongly related to the class. Now let's analyze the intuition discussed above with respect to one class, let's say `tech`, which contains a total of 401 documents. Figure 5.1 shows a heat map representation of all the document vectors belonging to `tech` class. In Figure 5.1, we are trying to investigate whether we can find the word clusters that might be strongly related to the class by looking at the word cluster with maximum non-zero weights throughout the documents of a class. In other words, we are interested in the axes for which most of the documents have non-zero components. To investigate this, the WcDe vector space axes on the x-axis are sorted in the decreasing order of the count of document in which the axis is hot (i.e. has a non-zero component). For the analysis, only the top 20 axes are shown in the heatmap. Additionally, to have the exact numbers for our discussion, Table 5.1 shows the exact statistics for the top 10 axes. For each axis, the sum, the maximum value, and the count of documents where the axis is hot ($\text{Count}_{>0}$) is indicated in the table. From Table 5.1 we can see that the top 5 axes are hot in all but 1 document of the class. However, the maximum component of these axes in all documents is 0.02 or less. Moreover, for the top 5 axes, the sum of all components corresponding to 400 documents is very small. This indicates that though the components along these axes are non-zero in nearly every document of `tech` class, the value of the components is really small, i.e. the weight of these word clusters is really small. The difference in the weights of the top 5 axes with the rest of the axes is much more apparent in Figure 5.1. It can be seen in Figure 5.1 that in comparison to clusters `BBC-281` and `BBC-1853`, the cluster weights of the top 5 axes (the leftmost 5 axes on the x-axis) are too small to even be discernible in the heatmap. When analyzing with respect to $\text{Count}_{>0}$ alone, this pattern is consistent for all the classes, as expected. Furthermore, on a closer look at the top 10 clusters in Table 5.2, we can see that most of the top 10 clusters are constituted of stopwords or words that are too common to add any distinctive value to the representation. Moreover, interestingly, in this heatmap, we can notice two axes labeled `BBC-281` and `BBC-1853` that show a clear pattern of large weights prevalent through most of the documents of this class. And these axes are exactly what we want to be at the top of our sorting order which is clearly not the case when sorting with respect to non-zero weight counts. So the criterion needs to be revisited.

Now, since the $\text{Count}_{>0}$ of axes, indicating the mere presence of non-zero value of the weight, didn't provide any insights into the search for document vector axes that may be contributing to capturing the similarity between the documents of the same class, let's rethink the criteria. The criterion of the mere presence of word clusters has proven to be insufficient. We've seen that the weights of the cluster are also an important part of our sorting criteria. Based on this we've identified two aspects of axes that are necessary for this investigation -

Prevalence - The first aspect of the axes is with respect to non-zero components in most documents of the class. I've called this aspect, the *prevalence* of the axis or the word cluster.

Strength - The second aspect of an axis that is relevant to the investigation is the value of its components in the documents of a class. This aspect is referred to as the *strength* of an axis in the document vectors of a class. In other words, it refers to the weight of the corresponding word cluster in the documents of a particular class.

The previous criteria that only considered the *prevalence* of an axis, was found to be insufficient for the analysis. In this investigation, we want to find the axes (the word clusters) that capture the similarity between the documents of the same class. And for that, the *strength* of the axes is at least as essential as prevalence. So let's consider a few other criteria with respect to whether they represent prevalence and strength of the axes, for example, maximum, sum, mean, median, variance, standard deviation of the distribution of the components of an axis throughout the document vectors. The

Cluster	**Members**
1714	the, of, in, for, on, with, by, from, an, over, part, between, under
578	is, that, it, was, has, have, are, they, this, which, had, been, were, also, now, being, still, already
2078	and, as, people, all, other, some, them, like, such, made, many, while, well, most, those, these, both, making, few, others, themselves
1892	to, make, take, need, able, hard, find, enough, needed, difficult, needs, bring, ways, finding, easier, unable, impossible, harder
218	at, up, one, out, time, only, just, back, next, way, home, off, down, going, go, where, half, put, day, come, place, another, away, every, times, past, each, point, here, coming, today, close, turn, twice
1814	be, will, not, would, can, if, could, should, may, does, must, might, cannot, unless
869	but, there, or, so, no, because, any, however, even, too, same, without, despite, far, seen, give, given, although, taking, yet, having, taken, clear, rather, though, saw, instead, either, gave, longer, giving, seeing, appears, thanks, upon
202	more, about, than, number, much, around, less, almost, least, total, numbers, nearly, estimated, fewer, scores, roughly, approximately
1522	their, its, my, our, own, your
717	year, after, last, years, before, since, week, months, during, month, early, earlier, recent, days, following, latest, ago, weeks, previous, late, followed, mid, initial, prior, shortly, subsequent

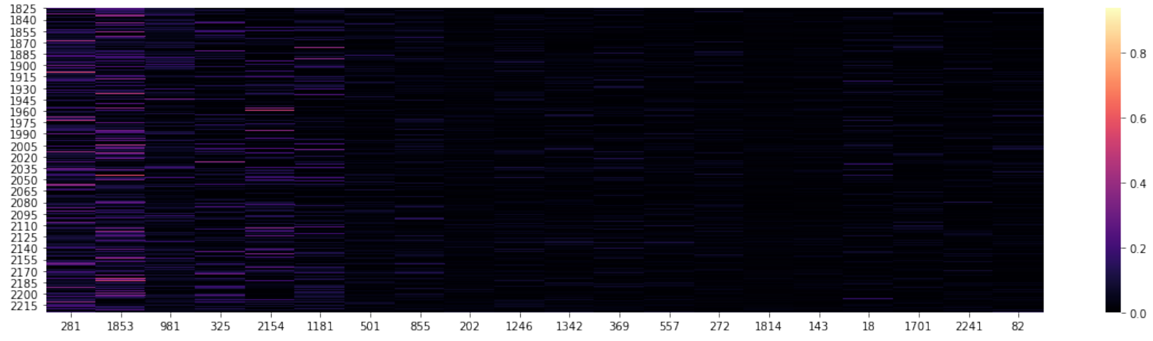
Table 5.2: Table listing word clusters and their members corresponding to the top 10 axes based on Count$_{>0}$ sorting of axes for `tech` class given in figure 5.1

variance and the standard deviation of the components of an axis (in the documents of a class), do say something about the variation of components with respect to the mean, but nothing about the mean value itself. Therefore, these criteria do not consider the strength of the axes at all. On the other hand, while the maximum weight of an axis in the documents of a class will represent the strength aspect, it does not consider the prevalence at all.

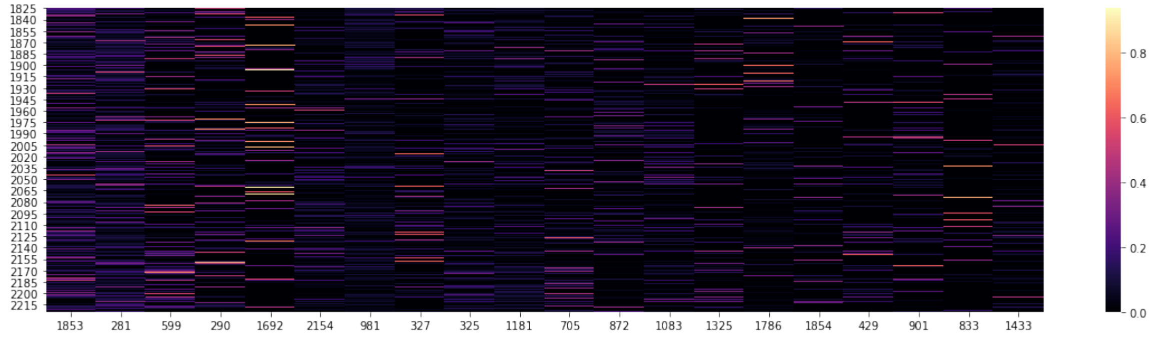
The median value of the components for an axis indicates the minimum component value for more than 50% of the documents. Therefore, a non-zero median indicates the prevalence of the axes in at least 50% of the document vectors, and the value of median indicates the minimum strength of the axis in 50% of the documents. In simpler words, median incorporates both prevalence and strength of axes to a degree. Similarly, the mean of components of an axis indicates the average strength of the axis throughout the documents of a class. Though numerically different, the mean and sum will both yield the same sort order for the axes, and therefore either will suffice for the analysis. Both the sum and the mean of components, will indicate a total strength of the axis based on all the documents of a class. This total is unlikely to be competitively high if the axis is not prevalent in the documents. Therefore, like median, both sum and mean of components also incorporate some consideration of the strength as well as the prevalence of the axes. Therefore, out of all the criteria that have been mentioned, it is only by considering the sum (or the mean) or the median, that we can go beyond the mere prevalence of axes, and in one way or another take the strength into account. Therefore, next, we'll explore the sum/mean and median values of components as criteria for investigating the axes that capture the intra-class similarity.

Figure 5.2 shows the heatmaps corresponding to top 20 clusters based on sum/mean and median values. While analyzing these heatmaps, what makes an axis interesting is not just its hotness on the given scale, but also the consistency of that hotness throughout a significantly skewed majority of documents in the cluster. In these heatmaps, we can see that the axes `281` and `1853` are now at the top of the sorting order. We can see that the first two axes (from the left) of both heatmaps, i.e. `281` and `1853`, are not only hot for most documents but are also consistently and significantly hotter than the other axes. Out of the two heatmaps, this is more apparent in Figure 5.2a, which is based on the median criteria. In Figure 5.2a, the strength of axes quickly fades out as we go from left to right. This indicates that while sorting based on the value of median indicates the minimum weight for a majority of documents, this is more in favour of prevalence than strength. On the other hand, when analyzing the axis sorted on the basis of the sum (Figure 5.2b), we can see multiple axes with very high weights, continuing all the way to the end of the heatmap. For additional analysis of the clusters represented by these axes, Table 5.3 shows the words that constitute the clusters corresponding to the top 10 axes sorted in the decreasing order of the sum of its weights. These axes are selected on the basis of documents of `tech` class only. Moreover, in Table 5.3 the words representing each cluster are the top 10 words selected in the decreasing order of frequency in the dataset.

In Table 5.3, we can see that the clusters are not only related to the `tech` class, but also demonstrate a strong relatedness between words within the cluster. For example, `BBC-290` contains the names of various organizations related to technology, `BBC-1692` contains words that are related



(a) Median



(b) Sum/Mean

Figure 5.2: Heatmap of top 20 axes of WcDe document vector representation of all documents of class `tech`. The axes have been sorted in the decreasing order of median (Figure 5.2a) or sum/mean (Figure 5.2b)

Cluster	Members
1853	digital, software, computer, devices, device, computers, portable, electronic, handheld, hardware
281	users, online, internet, content, web, entertainment, user, multimedia, interactive, programming
599	mobile, broadband, wireless, providers, subscribers, provider, voip, telephony, modem, subscriber
290	microsoft, apple, google, yahoo, ebay, aol, msn, skype, paypal, netscape
1692	nintendo, xbox, console, ds, consoles, playstation, psp, gameboy, ps2, sega
2154	technology, technologies, tech, innovation, inventions, technological, innovations
981	use, used, using, machine, machines, uses, tool, tools, equipment
327	pc, windows, mac, desktop, linux, xp, macintosh, os, symbian, solaris
325	service, services, system, systems, operating, operators, communications, operator, operate, operates
1181	phone, information, data, account, accounts, telephone, customer, clients, ups, client

Table 5.3: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `tech` class of BBC dataset

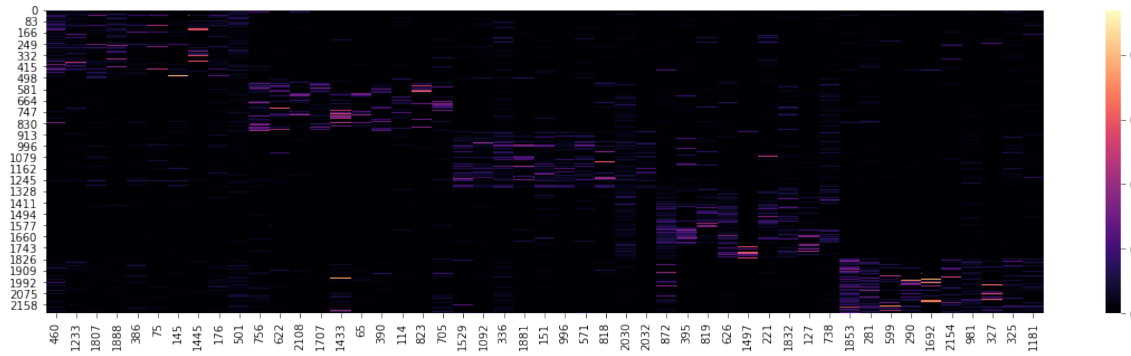
to digital games and `BBC-327` contains words that refer to computer systems or are names of operating systems. Just like Table 5.3 gives the top 10 words clusters and its members for `tech` class, Tables C.1 to C.10 in Appendix list top 10 word clusters for each class of both BBC datasets. Similar to the observations drawn from the top 10 word clusters of `tech` class, we can observe the same for the rest of the classes of both BBC datasets (Tables C.1 to C.10). This confirms that performance aside, to a reasonable extent, the WcDe methodology is able to capture what it was designed for - representation of documents in the terms of groups of similar words.

5.2 Captures Difference?

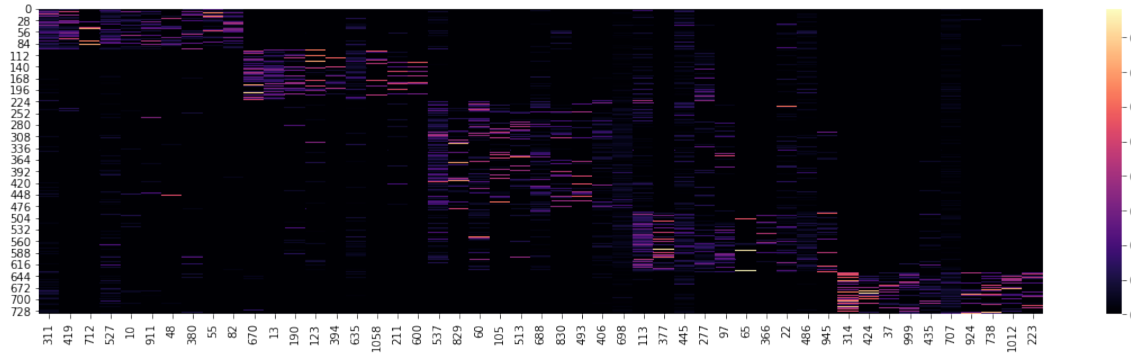
Based on the criterion of the sum of components, now we can find the axes that capture similarity for the documents of each class. So far we've investigated documents belonging to the same class, thus exhibiting a topical similarity. But what happens when we pitch the documents of different classes against each other? In order to analyze the documents with respect to the axes that most contribute to intra-class similarity and inter-class difference, I selected the top 10 axes (based on the sum of weights) from each class and plotted them together in a heatmap (Figure 5.3). So, on the x-axis of this heat map, we should have approximately 50 axes represented. However, for the axes that made the top 10 of more than one class, they were placed among the axes of the class where the axis ranked higher.

For a visual impact alone, in both of the heatmaps in Figure 5.3, the documents on the y-axis and the selected WcDe axes on the x-axis are both placed in the alphabetical order of classes. The same order of documents and axes allows the heatmap to depict clear-cut *boxes* indicating the strength and prevalence of the axes in the documents of the same class while clearly showing the difference in strength and prevalence in the documents of different classes. This contrast between the *boxes* and the area above and below them is indicative of the lack of overlap between the documents of different classes, and the ability of WcDe methodology to capture that. However, in Figure 5.3b, we can see that for the BBC Sport dataset, in the area on the top of the *rugby box*, the contrast is less, and the overlap is higher than that for the other classes. It is interesting to note that this shows that the axes selected for `rugby` class were strong in other classes as well. What is more interesting is that the class which exhibits strength in the axes selected for `rugby` class is `football`. Among all the classes of the BBC Sport dataset where each class refers to a sport, `rugby` and `football` are perhaps the most similar as sports. Overall, since the BBC Sport dataset contains articles with a common topic, i.e. `sport`, higher overlap of content and occurrence of similar word clusters is expected in the BBC Sport dataset than the BBC dataset.

It is clear from Figure 5.3 on top of Tables C.1 to C.10 that the selected axes are not only representative of intra-class similarity of the documents, but also represent the inter-class dissimilarity of the documents in both datasets. This entire analysis shows that the methodology of WcDe can represent the documents very well at the two levels of topical similarity without any loss in performance and interpretability while retaining a portion of participation of each word in the document.



(a) Heat map of document vectors of BBC dataset. The documents on the y-axis are originally ordered by sequence of class - `business`, `entertainment`, `politics`, `sport`, `tech`. The top 10 axes from each class are selected and placed on the x-axis in the same order as well.



(b) Heat map of document vectors of BBC Sport dataset. The documents on the y-axis are originally ordered by sequence of class - `athletics`, `cricket`, `football`, `rugby`, `tennis`. The top 10 axes from each class are selected and placed on the x-axis in the same order as well.

Figure 5.3: Heatmap of 50 axes of document vectors. Top 10 axes of each class have been selected in the decreasing order of their sum of weights. The axes have been arranged from left to right on the x-axis in the alphabetical sequence of class names.

Parameter		BBC	BBC Sport
Tokenization	Document Frequency Range	Relaxed: 1% - 95%	Relaxed: 1% - 95%
	Vector Size	2500	2500
	Stemming	Yes	No
Dimension of Topical Space		400	300

Table 5.4: LSA configuration selected for the analysis with respect to synonymy.

5.3 LSA and WcDe - Synonymy

At the beginning of Chapter 3, I raised the question of whether LSA is, in fact, able to address synonymy and polysemy. Since the experiments in this thesis are not attempting to address polysemy, I'll limit the analysis of both techniques with respect to the synonymy of words. For this analysis, the LSA configuration with best Normalized Mutual Information score was selected individually for both the datasets (Table 5.4). In LSA the term-document association data is broken down into associations of both the terms and documents with artificial concepts, i.e. topical axes. The association of a document with these artificial concepts is then used as a document vector. Similarly, the association of these artificial concepts with the terms in the dataset can be treated as a vector representation of words in terms of topical axes. This association data can also be interpreted as the distribution of terms over the artificial concepts and therefore could be considered a definition of an artificial concept, i.e. a topic, as a distribution over terms. Therefore, in order to investigate the artificial vector space generated by Latent Semantic Analysis, we can look at the term distribution for each axis.

In Latent Semantic Analysis, due to the decomposition using Singular-Value Decomposition, the topical axes are arranged in the order of singular values such that, the axes with the lowest singular values can be removed while minimizing the discrepancy between the data before and after removal of axes. Therefore, in order to investigate to what extent the topical axes are able to actually address the synonymy of words, I've selected the topical axes based on singular values. For this discussion, Table 5.5 shows only the top 10 axes in the decreasing order of the singular values in the LSA vector space for the BBC dataset. Since the topic vector is a distribution over terms, based on the weight of each term in the topic vector, we can obtain the terms that strongly associate with the topic. For the purpose of this discussion, (Table 5.5) shows only the top 5 terms and their association score with each of the top 10 topical axes. However, you can refer to Tables C.11 and C.12 in the Appendix which shows top 20 topic axes for LSA vector space for both datasets.

In Table 5.5, we can see that the terms associated with `LSA-BBC-7` are all related to music, the ones associated with `LSA-BBC-4` are mostly related to movies, and the ones with `LSA-BBC-5` are related to technology. However, there is less relatedness to be observed within the terms associated with a topical axis. In comparison to others, `LSA-BBC-4` , `LSA-BBC-5` and `LSA-BBC-7` bear more semblance of a topic. This is even more difficult to observe for topical axes for the BBC Sport

#	Associated Terms
1	said (0.22), mr (0.16), year (0.13), would (0.11), game (0.11)
2	mr (0.27), labour (0.2), elect (0.2), blair (0.16), parti (0.16)
3	labour (0.17), elect (0.16), blair (0.15), mr (0.14), parti (0.14)
4	film (0.49), award (0.28), best (0.21), star (0.16), oscar (0.16)
5	mobil (0.22), phone (0.2), use (0.17), peopl (0.15), technolog (0.15)
6	yuko (0.18), court (0.14), compani (0.14), firm (0.13), law (0.13)
7	music (0.3), band (0.25), album (0.23), chart (0.18), song (0.15)
8	game (0.31), club (0.17), music (0.14), unit (0.13), sale (0.13)
9	england (0.26), music (0.19), wale (0.19), band (0.17), album (0.14)
10	mobil (0.4), phone (0.32), club (0.11), england (0.1), handset (0.09)

Table 5.5: List of terms (stemmed) and their association with each of the top 10 axes of LSA vector space on BBC Dataset. The top 10 axes have been selected in the decreasing order of their singular values.

dataset in Table C.12. Moreover, it is worth noting that even if the strongly associated terms do bear any topical semblance, there is hardly any synonymy to be observed.

Furthermore, although it makes less sense to do so for LSA, if I apply the same methodology for selecting topical axes as I applied for WcDe in the previous section, I can get some topical axes that seem to bear more relation to the class. In other words, for each class in the dataset, the top 10 topical axes are selected based on the sum of its weights across the documents of the class. For example, Table 5.6 gives the top 5 axes obtained based on the sum of weights of documents of `tech`. For both, LSA and WcDe, axes were selected on the basis of the sum of component criteria we discussed in the previous section. For LSA, the top 5 terms, based on their term-topic association score, represent each topical axis in the table. For WcDe, the axes are represented by the top 5 words from the corresponding word cluster based on their frequency in the dataset. Table 5.6 only provides the comparison for `tech`. However, the same comparison for the remaining class of both, BBC and BBC Sport, dataset are available for reference in the Appendix (Tables C.13 and C.14).

Axis Rank	LSA	WcDe
1	samsung, suppos, procedur, exceed, assum	digital, software, computer, devices, device
2	economi, growth, bank, rate, econom	users, online, internet, content, web
3	film, seed, mr, festiv, blair	mobile, broadband, wireless, providers, subscribers
4	tax, parti, film, tori, yuko	microsoft, apple, google, yahoo, ebay
5	india, site, mobil, search, blog	nintendo, xbox, console, ds, consoles

Table 5.6: A comparison between the top 5 topics/clusters (represented by 5 terms/words each) of both LSA and WcDe vector spaces. The top 5 topics/clusters were selected on the basis of sum of weights across the document vectors of `tech` class.

From Table 5.6, the first and the fifth ranking axes of LSA bear some relation to the topic of technology. But as is clear from the comparison, the word clusters from WcDe demonstrate more coherence within the cluster and even within the class (Tables C.13 and C.14).

5.4 Concluding Observations

Finally, to conclude the qualitative analysis of WcDe, let's wrap up with some observations based on the cluster makeup of the few clusters listed in given in Tables C.1 to C.10 -

1. Some clusters, not all, contain terms that can be considered synonymous. For example -

Cluster #	**Members**	**Near Synonyms**
`BBC-1832`	win, won, winning, victory, wins, victories	win, victory
`BBC-622`	awards, award, prize, honour, awarded, prizes, honorary, merit, honor, awarding	award, prize, honor, merit

2. Though many clusters don't entirely contain synonymous words, most of these clusters contain words that are used in very similar contexts. For example, there are clusters containing all genres, and others containing names of organizations and people.

Cluster #	**Members**	**Concept**
`BBC-390`	comedy, drama, sequel, thriller, horror, biopic, remake, sitcom, spoof, parody, flick, westerns, satire, sitcoms, comedies, dramas, suspense	Genre

Interestingly, though both might be related to `tech`, it also separate the names of organizations from the names of softwares and services -

Cluster #	**Members**	**Common Theme**
`BBC-290`	microsoft, apple, google, yahoo, ebay, aol, msn, skype, paypal, netscape, hotmail, gmail	Organizations / Online Services
`BBC-327`	pc, windows, mac, desktop, linux, xp, macintosh, os, symbian, solaris	Computer Machines / Operating Systems

Furthermore, the clusters not only separate the names of the players on the basis of the sport, but also separates the names of the players participating on behalf of different countries -

Cluster #	Members	Common Theme
`BBC-SPORT-493`	ronaldo, beckham, ronaldinho, juninho, adriano, zidane, cristiano, zinedine, kaka, rivaldo, figo, robinho	Football Players
`BBC-SPORT-924`	kuznetsova, sharapova, dementieva, svetlana, petrova, davydenko, youzhny, nikolay, bondarenko, zvonareva, andreev, dinara, safina	Tennis Players (Russia)
`BBC-SPORT-600`	ntini, steyn, nel, makhaya, morkel, charl, langeveldt, albie	Cricket Players (South Africa)
`BBC-SPORT-1058`	shoaib, akhtar, kaneria, razzaq, akmal, kamran, umar, shahid, afridi, mushtaq, shabbir	Cricket Players (Pakistan)

3. Using WcDe for document representation obviates the need for some text processing like, stemming and lemmatization because the different words related to the underlying concept are clustered together in the same cluster. Consider the italicized words in "Members" column and the corresponding entry in the "Common" column in the table given below -

Cluster #	Members	Common
`BBC-705`	tv, radio, network, networks, television, channel, *broadcast*, cable, station, *broadcasting*, stations, channels, *broadcasters*, *broadcaster*, *broadcasts*	broadcast
`BBC-386`	growth, *increase*, *increased*, *increasing*, output, *increases*, reduced, gdp, gross, consumption, contraction, decrease, decreases	increase
`BBC-818`	*vote*, *voters*, poll, *voting*, *votes*, ballot, polls, *voter*, turnout, polling, ballots, counting, counted, balloting, absentee	vote
`BBC-1832`	*win*, *won*, *winning*, victory, *wins*, victories	win
`BBC-SPORT-635`	*test*, *tests*, *testing*, *tested*, samples, sample, urine	test

Chapter 6

Conclusion

The explosion of textual data on the internet has long surpassed the capabilities of manual sorting and processing. The effective automation of processing such a large amount of data, much of which is usually unlabelled, usually requires that it be represented as a mathematical object or vector. As we've seen in Chapter 4, WcDe methodology outperforms the existing popular unsupervised textual document representation techniques. Moreover, it consists of various modular components, i.e. word vectors, clustering technique, and weighting function, that can be adjusted according to the need of the task. Furthermore, as we've seen in Section 4.4.1, an analysis of the components narrows down the variation in performance while maintaining the quality. The insight on the best option for each component and a robust variation is particularly useful for unsupervised tasks because in unsupervised tasks there is no output label that can be used to adjust or compute the model weights. Moreover, the combination of components so found can be observed to perform consistently well across different datasets and at different levels of topical similarity. Not to mention the WcDe methodology allows the representation of document without completely discarding any words in the vocabulary (unlike Vector Space Model) while taking the similarity of words into account (unlike Latent Semantic Analysis) and maintaining interpretability of features (unlike Doc2vec) without compromising on performance at either level of similarity. That said, there are multiple avenues of research and further development of this technique. Some of these avenues are discussed below.

6.1 Future Work

A set of future work in WcDe methodology can correspond to further development and investigation of each of the individual components, the first of the components being the word embeddings or word vectors. The word embeddings used in the experiments discussed in this thesis are non-contextualized word embeddings, i.e. there is a single word vector corresponding to a word and the context in which the word appears is not taken into account while generating word representation. But, a single word can be used to convey different meanings, and therefore, a single vector representation of a word may not suffice for words that represent different meanings in different contexts. Therefore, non-contextualized embeddings do not address the polysemy of words in the textual language data.

However, in another type of word embeddings, called *contextualized word embeddings*, the context of the word is taken into account while obtaining the word vector for a word. WcDe method addresses synonymy of words but by using contextualized word embedding like BERT, it can also address polysemy which has remained unaddressed in WcDe so far. Therefore, one of the major directions for future work in further development of WcDe methodology is with respect to the use of contextualized word embeddings.

Moreover, in this thesis, the experiments were limited to hard clustering techniques. But, as we've just discussed regarding the addressal of polysemy of words, words can belong to more than one cluster, and to different degrees. Therefore, by using soft clustering techniques for clustering word vectors, we can define a word's membership as a distribution over word clusters. This is similar to the definition of words in terms of artificial concepts in Document-Topic Vector Space Model. In other words, partial membership of words in word clusters can be investigated for its potential to improve the representation of documents.

Moreover, in the experiments discussed in this thesis, the vocabulary of the word vectors was limited to the vocabulary of the dataset. Another avenue of investigation involves expanding the vocabulary beyond the dataset and finding a universal set of word clusters that can be used to represent the documents effectively. This might result in very high dimensionality of WcDe document vector space. But, using an analysis similar to the Document-Term Vector Space Model, the word clusters with document frequency below a pre-decided threshold can be discarded. Therefore, another interesting avenue of investigation is of the dimensionality reduction of WcDe vectors. In conclusion, while some aspects of Word Cluster based Document Embedding were investigated in this thesis, there are some promising avenues of investigation for the future work as well.

References

Agirre, Eneko, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe (June 2015). "SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, pp. 252–263.

Agirre, Eneko, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe (Aug. 2014). "SemEval-2014 Task 10: Multilingual Semantic Textual Similarity". In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, pp. 81–91.

Agirre, Eneko, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre (June 2012). "SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity". In: **SEM 2012: 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 (SemEval 2012)*. Montréal, Canada: Association for Computational Linguistics, pp. 385–393.

Agirre, Eneko, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo (June 2013). "*SEM 2013 shared task: Semantic Textual Similarity". In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. Atlanta, Georgia, USA: Association for Computational Linguistics, pp. 32–43.

Aldous, David J. (1985). "Exchangeability and related topics". In: *École d'Été de Probabilités de Saint-Flour XIII — 1983*. Ed. by P. L. Hennequin. Vol. 117. Springer Berlin Heidelberg, pp. 1–198.

Arora, Sanjeev, Yingyu Liang, and Tengyu Ma (2017). "A simple but tough-to-beat baseline for sentence embeddings". In: *5th International Conference on Learning Representations (ICLR)*.

Arthur, David and Sergei Vassilvitskii (2007). "K-means++: The advantages of careful seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, pp. 1027–1035.

Becker, Hila (2011). "Identification and characterization of events in social media". PhD dissertation. Columbia University.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, Christian Jauvin, Jauvinc@iro Umontreal Ca, Jaz Kandola, Thomas Hofmann, Tomaso Poggio, and John Shawe-Taylor (2003). "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3, pp. 1137–1155.

Blei, David M, Andrew Y Ng, and Michael I. Jordan (2003). "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3, pp. 993–1022.

Blei, David M. (2012). "Probabilistic topic models". In: *Communications of the ACM* 55.4, pp. 77–84.

Cover, Thomas M and Joy A Thomas (2005). "Entropy, Relative Entropy, and Mutual Information". In: *Elements of Information Theory.* John Wiley & Sons, Ltd. Chap. 2, pp. 12–49.

Dai, Xiangfeng, Marwan Bikdash, and Bradley Meyer (2017). "From social media to public health surveillance: Word embedding based clustering method for twitter classification". In: *Southeast-Con 2017*, pp. 1–7.

De Boom, Cedric, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt (Sept. 2016). "Representation Learning for Very Short Texts Using Weighted Word Embedding Aggregation". In: *Pattern Recognition Letters* 80.C, pp. 150–156.

Deerwester, Scott, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman (1990). "Indexing by latent semantic analysis". In: *Journal of the American Society for Information Science* 41.6, pp. 391–407.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186.

Firth, John R (1957). "A synopsis of linguistic theory, 1930-1955". In: *Studies in Linguistic Analysis.* Oxford, pp. 1–32.

Fowlkes, E B and C L Mallows (1983). "A method for comparing two hierarchical clusterings". In: *Journal of the American Statistical Association* 78.383, pp. 553–569.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). "Linear Algebra". In: *Deep Learning.* MIT Press. Chap. 2, pp. 29–50.

Hoffman, Matthew D, David M Blei, and Francis Bach (2010). "Online learning for Latent Dirichlet Allocation". In: *Proceedings of the 23rd International Conference on Neural Information Processing Systems*. Vol. 1.

Hubert, Lawrence and Phipps Arabie (1985). "Comparing partitions". In: *Journal of Classification* 2.1, pp. 193–218.

Jurafsky, Daniel and James H Martin (2021). "Vector Semantics and Embeddings". In: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* 3rd ed. (draft of December 30, 2020). Chap. 6.

Kim, Han Kyul, Hyunjoong Kim, and Sungzoon Cho (2017). "Bag-of-concepts: Comprehending document representation through clustering words in distributed representation". In: *Neurocomputing* 266, pp. 336–352.

Le, Quoc and Tomas Mikolov (2014). “Distributed Representations of Sentences and Documents”. In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, pp. 1188–1196.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008a). *Flat Clustering*. Cambridge University Press. Chap. 16, pp. 349–376.

— (2008b). *Hierarchical Clustering*. Cambridge University Press. Chap. 17, pp. 377–402.

— (2008c). *Introduction to Information Retrieval*. Cambridge University Press, p. 505.

— (2008d). *Matrix decompositions and latent semantic indexing*. Cambridge University Press. Chap. 18, pp. 403–420.

Mekala, Dheeraj, Vivek Gupta, Bhargavi Paranjape, and Harish Karnick (2017). *SCDV: Sparse Composite Document Vectors using soft clustering over distributional representations*. Tech. rep., pp. 659–669.

Mikolov, Tomás, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). “Efficient Estimation of Word Representations in Vector Space”. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018). “Deep contextualized word representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237.

Qimin, Cao, Guo Qiao, Wang Yongliang, and Wu Xianghua (2015). “Text clustering using VSM with feature clusters”. In: *Neural Computing and Applications* 26.4, pp. 995–1003.

Reynolds, Douglas (2015). “Gaussian Mixture Models”. In: *Encyclopedia of Biometrics*. Ed. by Stan Z Li and Anil K Jain. Boston, MA: Springer US, pp. 827–832.

Richards, Blake A, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, Colleen J. Gillon, Danijar Hafner, Adam Kepecs, Nikolaus Kriegeskorte, Peter Latham, Grace W. Lindsay, Kenneth D. Miller, Richard Naud, Christopher C. Pack, Panayiota Poirazi, Pieter Roelfsema, João Sacramento, Andrew Saxe, Benjamin Scellier, Anna C. Schapiro, Walter Senn, Greg Wayne, Daniel Yamins, Friedemann Zenke, Joel Zylberberg, Denis Therien, and Konrad P. Kording (2019). “A deep learning framework for neuroscience”. In: *Nature Neuroscience* 22.11, pp. 1761–1770.

Rong, Xin (2014). “word2vec Parameter Learning Explained”. In: *CoRR* abs/1411.2738.

Rosenberg, Andrew and Julia Hirschberg (June 2007). “V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure”. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 410–420.

Seifollahi, Sattar, Massimo Piccardi, Ehsan Zare Borzeshi, and Bernie Kruger (2019). “Taxonomy-Augmented Features for Document Clustering”. In: *Australasian Conference on Data Mining (AusDM): Data Mining*. Ed. by Rafiqul Islam, Yun Sing Koh, Yanchang Zhao, Graco Warwick, David Stirling, Chang-Tsun Li, and Zahidul Islam. Communications in Computer and Information Science (CCIS), Vol. 996. Springer Singapore, pp. 241–252.

Socher, Richard (2015). *CS 224 D : Deep Learning for NLP (Lecture Notes: Part I)*. Available at https://cs224d.stanford.edu/lecture_notes/LectureNotes1.pdf (2021/08/03).

Wikimedia (2012). *English Wikipedia dump*. Available at http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2 (2021/02/06).

Wikipedia contributors (2021). *Mutual information – Wikipedia, The Free Encyclopedia*. Available at https://en.wikipedia.org/w/index.php?title=Mutual_information&oldid=1035953322 (2021/08/03).

Appendix A

Quantitative Evaluation - Additional Plots

This chapter provides additional boxplots for reference with respect to the discussion in Evaluation & Quantitative Analysis. Figures A.1 to A.7 show boxplots of distribution of scores for each method for different evaluation metrics.

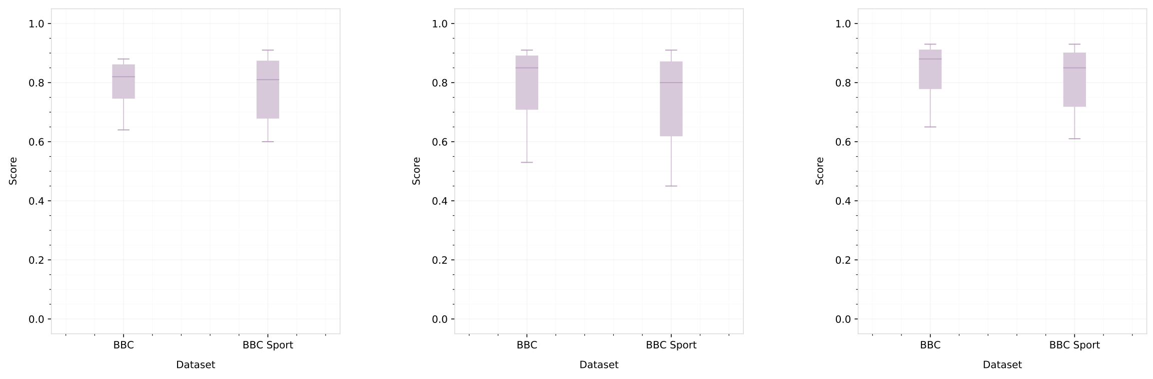


Figure A.1: Boxplots depicting the distribution of various scores over DT-VSM experiments on each dataset.

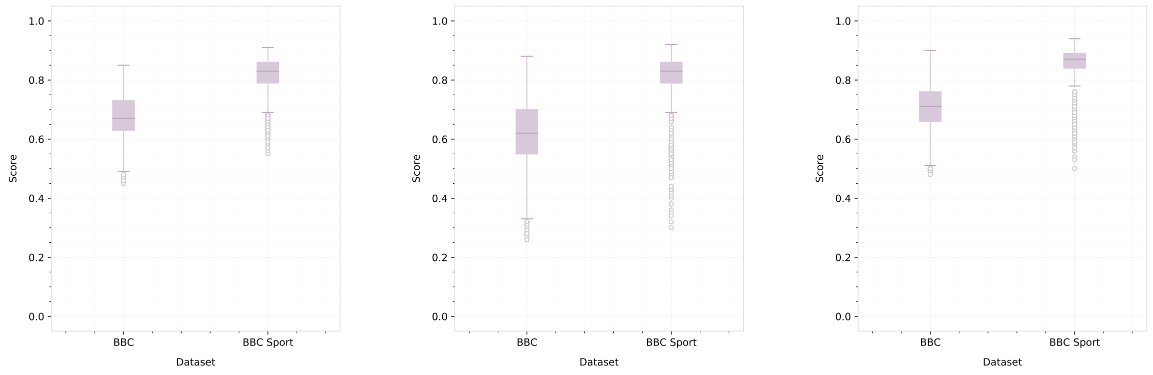


Figure A.2: Boxplots depicting the distribution of various scores over LSA experiments on each dataset.

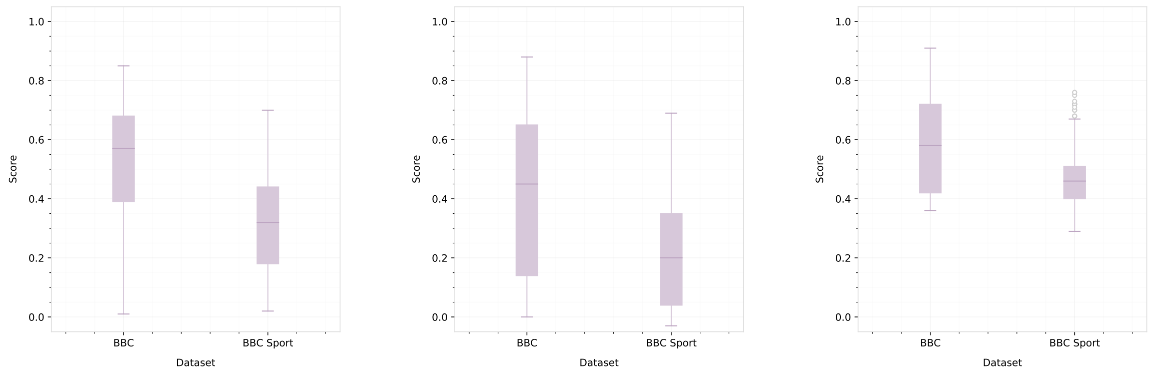


Figure A.3: Boxplots depicting the distribution of various scores over LDA experiments on each dataset.

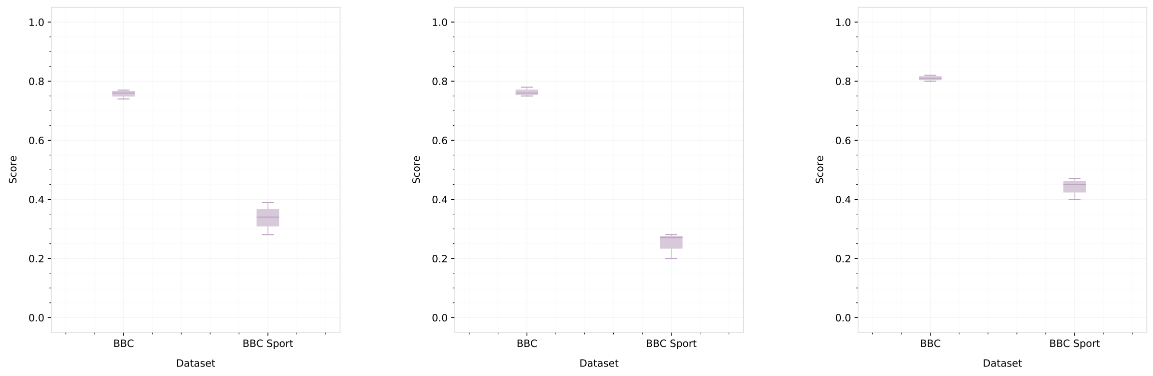


Figure A.4: Boxplots depicting the distribution of various scores over `WE_AVG` experiments on each dataset.

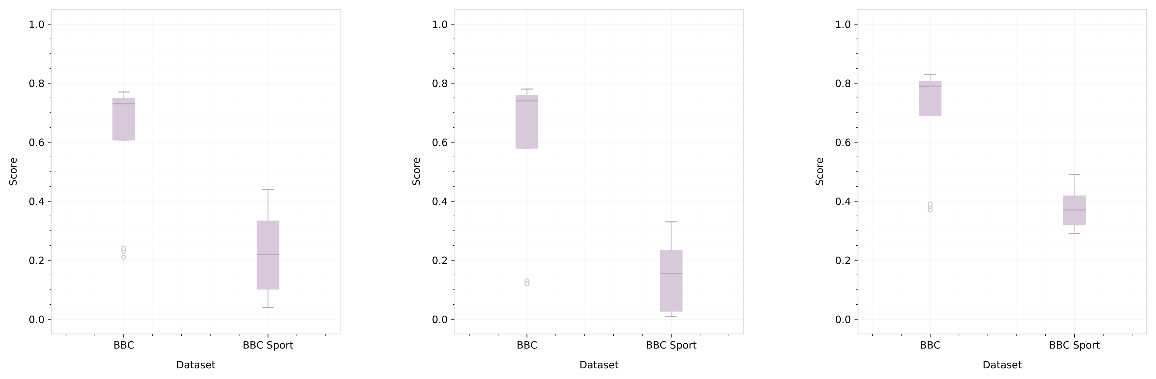


Figure A.5: Boxplots depicting the distribution of various scores over WE_TFIDF experiments on each dataset.

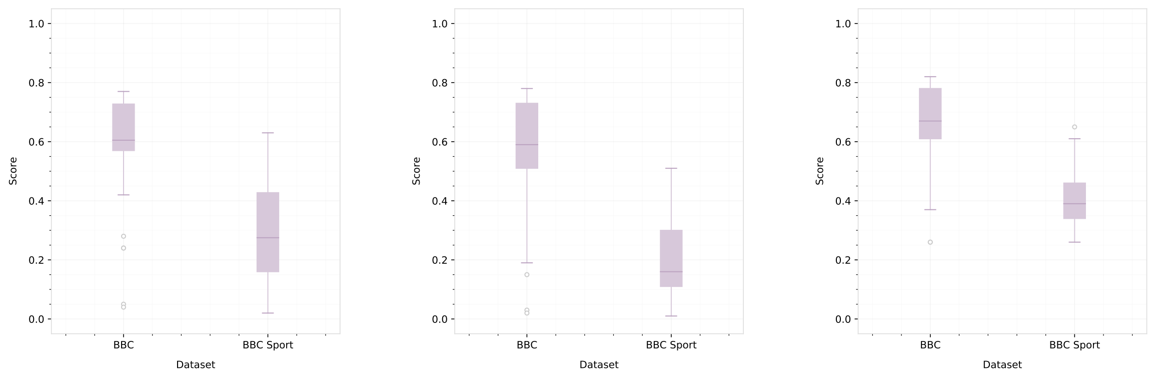


Figure A.6: Boxplots depicting the distribution of various scores over WE_SIF experiments on each dataset.

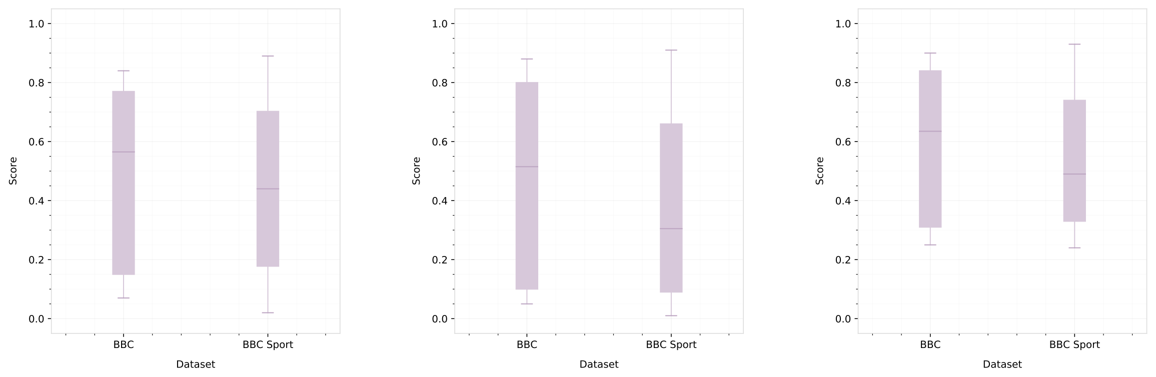


Figure A.7: Boxplots depicting the distribution of various scores over Doc2vec experiments on each dataset.

Appendix B

Quantitative Evaluation - Additional Tables

This chapter provides additional tables for reference with respect to the discussion in Evaluation & Quantitative Analysis. Table B.1 gives the performance scores distribution of WcDe experiments with respect to different word embeddings. The experiments given in Table B.1 use either K-Means or AHC with minimum variance merge criterion for clustering the word vectors. The document vector is generated by applying CF-iDF weight function and applying length normalization to the resulting vector. The cells in Table B.1 are color-coded with a distribution of color from red to green with the median value of color distribution fixed at 0.75. The median at 0.75 results in nearly no color at values that are close to 0.75. The lower the value is from 0.75 the darker the red color of the cell is, and the higher the value from 0.75, the darker the green color of the cell is.

		BBC		BBC Sport	
Percentile	**Word Embedding**	**K-Means**	**AHC ward**	**K-Means**	**AHC ward**
Min	GloVe 100d	0.69	0.65	0.65	0.71
	GloVe 300d	0.71	0.67	0.57	0.70
	Word2vec Pre-trained	0.70	0.67	0.47	0.42
	Word2vec 20 Epochs	0.78	0.69	0.67	0.67
	Word2vec 50 Epochs	0.61	0.55	0.63	0.58
	Word2vec 100 Epochs	0.65	0.54	0.52	0.55
	Word2vec 150 Epochs	0.59	0.49	0.54	0.63
25th	GloVe 100d	0.71	0.71	0.74	0.77
	GloVe 300d	0.79	0.71	0.68	0.73
	Word2vec Pre-trained	0.74	0.74	0.59	0.51
	Word2vec 20 Epochs	0.84	0.76	0.80	0.73
	Word2vec 50 Epochs	0.71	0.66	0.72	0.76
	Word2vec 100 Epochs	0.72	0.60	0.70	0.74
	Word2vec 150 Epochs	0.63	0.59	0.67	0.72
50th	GloVe 100d	0.75	0.76	0.78	0.79
	GloVe 300d	0.81	0.75	0.79	0.75
	Word2vec Pre-trained	0.77	0.75	0.61	0.57
	Word2vec 20 Epochs	0.85	0.81	0.84	0.74
	Word2vec 50 Epochs	0.78	0.72	0.76	0.80
	Word2vec 100 Epochs	0.75	0.64	0.77	0.76
	Word2vec 150 Epochs	0.74	0.63	0.71	0.76
75th	GloVe 100d	0.79	0.83	0.85	0.82
	GloVe 300d	0.82	0.81	0.85	0.76
	Word2vec Pre-trained	0.79	0.77	0.73	0.65
	Word2vec 20 Epochs	0.86	0.85	0.86	0.82
	Word2vec 50 Epochs	0.83	0.74	0.81	0.82
	Word2vec 100 Epochs	0.79	0.75	0.78	0.78
	Word2vec 150 Epochs	0.78	0.65	0.76	0.82
Max	GloVe 100d	0.82	0.85	0.88	0.89
	GloVe 300d	0.86	0.83	0.89	0.84
	Word2vec Pre-trained	0.81	0.85	0.79	0.75
	Word2vec 20 Epochs	0.89	0.88	0.90	0.87
	Word2vec 50 Epochs	0.86	0.83	0.90	0.87
	Word2vec 100 Epochs	0.83	0.80	0.87	0.83
	Word2vec 150 Epochs	0.82	0.77	0.87	0.89

Table B.1: The tables give the distribution of performance scores over WcDe experiments where the word vectors are clustered with either K-Means or AHC with minimum variance merge criterion and the document vectors are created using CF-iDF weight function and length normalized in the end. The median of color distribution is fixed to be 0.75.

Appendix C

Document Vector Axes - LSA and WcDe

This chapter provides additional tables for Discussion & Qualitative Analysis -

- Tables C.1 to C.10 lists top 10 axes and their corresponding word clusters for each class of BBC and BBC Sport datasets. The top 10 axes are selected on the basis of sum of components. The words listed as the members of the word cluster are listed in the decreasing order of frequency in the dataset.

- Tables C.11 and C.12 list out the top 20 axes for LSA artificial vector space for the LSA experiments discussed in Section 5.3. For both datasets Tables C.11 and C.12 lists the axes selected on the basis of singular values. The axes have been sorted in the decreasing order of their singular values. The axes here are indicative of an artificial concept. The artificial concept is a distribution over terms and for each axis, the tables provide a list of associated terms and their weights with respect to the artificial concept. Only the terms with a weight of 0.1 or more have been listed in the tables.

- Tables C.13 and C.14 provide a comparison of LSA topics and WcDe word clusters for each class of both BBC and BBC Sport datasets. Just like the sum criterion for the analysis of WcDe document vectors in Chapter 5, for each artificial axes of LSA document vectors, their components are were summed for the documents of the class. Therefore, the LSA topical axes in this table are sorted in the decreasing order of this sum. The table only lists the top 5 topical axes. For WcDe as well, the axes are selected on the basis of the sum criterion. The top 5 terms are selected to represent each axis of either method.

Axis	Members
460	market, sales, prices, demand, consumer, price, investors, markets, retail, traders
1233	bank, financial, investment, credit, banks, corporate, assets, securities, banking, investments
1807	economy, economic, global, asia, asian, economies, emerging
1888	shares, share, stock, exchange, trading, gains, stocks, futures, traded
386	growth, increase, increased, increasing, output, increases, reduced, gdp, gross, consumption
75	rate, rates, unemployment, jobless, hike, joblessness, hikes
145	spending, budget, deficit, fiscal, deficits, budgets, surplus, expenditure, shortfall, surpluses
1445	oil, crude, exploration, petroleum, offshore, drilling, refining, refinery, refineries, rigs
176	figures, analysts, figure, predicted, forecast, forecasts, economists, estimates, outlook, predict
501	company, firm, industry, based, business, enterprise, consultant, consultancy, consulting

Table C.1: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `business` class of BBC dataset

Axis	Members
756	film, films, series, movie, hollywood, movies, documentary, scene, episode, scenes
622	awards, award, prize, honour, awarded, prizes, honorary, merit, honor, awarding
2108	band, rock, pop, hip, hop, rap, bands
1707	actor, actress, starring, starred
1433	music, artists, musical, artist, musicians, performers, singers, songwriters, composers
65	album, song, label, albums, labels, demos, unreleased, demo, reissued, reissues
390	comedy, drama, sequel, thriller, horror, biopic, remake, sitcom, spoof, parody
114	festival, performed, concert, perform, performing, gig, concerts, gigs, touring, toured
823	swank, dicaprio, hilary, winslet, hanks, blanchett, depp, cate, spacey, gere
705	tv, radio, network, networks, television, channel, broadcast, cable, station, broadcasting

Table C.2: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `entertainment` class of BBC dataset

Axis	Members
1529	election, parliament, elections, parliamentary, assembly, electoral, legislative
1092	party, leader, political, opposition, politics, leadership, presidency
336	mr, dr, mrs, prof, baroness, portman, firstly, secondly, thirdly, lynda
1881	blair, downing, gates, colin, pentagon, powell, ames, condoleezza
151	labour, union, association, unions, federation, labor, federations, confederation, associations
996	tory, tories, backbenchers, backbencher, eurosceptic, backbench, sceptic, backbenches, frontbench, eurosceptics
571	minister, prime, defence, premier, interior, defense
818	vote, voters, poll, voting, votes, ballot, polls, voter, turnout, polling
2030	he, his, she, her, him, himself, herself
2032	liberal, conservative, conservatives, moderate, libertarian, liberals, progressive, leaning

Table C.3: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `politics` class of BBC dataset

Axis	Members
872	game, games, players, team, player, season, coach, teams, coaches, coaching
395	england, wales, ireland, scotland, irish, scottish, english, welsh, scots
819	chelsea, liverpool, manchester, newcastle, leicester, birmingham, sheffield, everton, southampton, leeds
626	final, match, round, champions, matches, tournament, championship, finals, uefa, qualifying
1497	roddick, hewitt, nadal, federer, henman, agassi, safin, rusedski, lleyton, marat
221	club, league, football, sport, sports, clubs, class, professional, sporting, junior
1832	win, won, winning, victory, wins, victories
127	injury, leg, injuries, knee, ankle, hamstring, shoulder, thumb, wrist, groin
738	williams, jones, gordon, robinson, smith, davis, johnson, campbell, taylor, jackson
2030	he, his, she, her, him, himself, herself

Table C.4: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `sport` class of BBC dataset

Axis	Members
1853	digital, software, computer, devices, device, computers, portable, electronic, hand-held, hardware
281	users, online, internet, content, web, entertainment, user, multimedia, interactive, programming
599	mobile, broadband, wireless, providers, subscribers, provider, voip, telephony, modem, subscriber
290	microsoft, apple, google, yahoo, ebay, aol, msn, skype, paypal, netscape
1692	nintendo, xbox, console, ds, consoles, playstation, psp, gameboy, ps2, sega
2154	technology, technologies, tech, innovation, inventions, technological, innovations
981	use, used, using, machine, machines, uses, tool, tools, equipment
327	pc, windows, mac, desktop, linux, xp, macintosh, os, symbian, solaris
325	service, services, system, systems, operating, operators, communications, operator, operate, operates
1181	phone, information, data, account, accounts, telephone, customer, clients, ups, client

Table C.5: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `tech` class of BBC dataset

Axis	Members
311	olympic, championships, event, competition, athletics, athletes, olympics, events, athlete, competitions
419	60m, 200m, 100m, 1500m, 800m, 400m, 50m, 500m, 3000m, 1000m
712	kenteris, thanou, katerina, annus
527	race, grand, finish, finished, runners, runner, finishing, prix, races, pole
10	gold, medal, silver, medals, bronze, bronzes, golds, silvers
911	athens, greek, greece, cyprus, turkey, turkish, cypriot
48	iaaf, doping, blatter, usada, ioc, cas, wada, sepp, usatf, rogge
380	indoor, indoors, outdoor, pool, makeshift, playground, outdoors, backyard, pools
55	chepkemei, kluft, heptathlon, gatlin, masai, lagat, kelli, kiplagat, obikwelu, joyner
82	marathon, sprinters, hurdles, sprint, marathons, sprinting, hurdle, sprints, obstacle, hurdlers

Table C.6: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `athletics` class of BBC Sport dataset

Axis	Members
670	pakistan, india, sri, bangladesh, lanka, lankan
13	wicket, overs, wickets, batsman, bowler, bowled, bowling, batsmen, fours, bowlers
190	vaughan, trescothick, giles, gillespie, thorpe, gough, hogg, cairns, fleming, solanki
123	ponting, martyn, lehmann, hayden, warne, gilchrist, langer, damien, mcgrath, katich
394	pietersen, flintoff, harmison, hoggard, lbw, collingwood, nought, caddick
635	test, tests, testing, tested, samples, sample, urine
1058	shoaib, akhtar, kaneria, razzaq, akmal, kamran, umar, shahid, afridi, mushtaq
277	rugby, cricket, fifa, icc, uefa, sevens, irb
211	pollock, boje, graeme, shaun, jaarsveld, cronje, ontong, hansie, rhodes, jonty
600	ntini, steyn, nel, makhaya, morkel, charl, langeveldt, albie

Table C.7: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `cricket` class of BBC Sport dataset

Axis	Members
537	chelsea, liverpool, manchester, newcastle, everton, leicester, birmingham, southampton, sheffield, blackburn
829	arsenal, wenger, arsene, gunners
60	gerrard, ferguson, hodgson, owen, wilkinson, rooney, bellamy, lampard, giggs, jonny
105	mourinho, benitez, aragones, rafa, houllier, mcleish, ranieri, barca, cruyff, rijkaard
513	souness, redknapp, keegan, moyes, eriksson, mcclaren, jol, hoddle, gatland, warnock
688	goal, scored, goals, score, scoring
830	madrid, barcelona, villa, palace
493	ronaldo, beckham, ronaldinho, juninho, adriano, zidane, cristiano, zinedine, kaka, rivaldo
406	premiership, fa, trophy, fixture, trophies, fixtures, silverware
698	team, players, club, coach, league, player, football, teams, clubs, coaches

Table C.8: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `football` class of BBC Sport dataset

Axis	Members
113	england, ireland, wales, scotland, irish, english, edinburgh, welsh, scottish, dublin
377	wasps, dallaglio, saracens, corry, vickery, grewcock, worsley, borthwick, harlequins
445	jones, williams, robinson, smith, johnson, holmes, cole, taylor, davis, lewis
277	rugby, cricket, fifa, icc, uefa, sevens, irb
97	lions, rangers, saints, wolves, dragons, sharks, reds, tigers, falcons, warriors
65	easterby, hickie, humphreys, horgan, ronan, maggs, horan, sheahan, leamy, geordan
366	luscombe, shanklin, sidoli, yapp, cockbain, mcbryde, charvis, gethin, voyce, ceri
22	henson, dawson, connor, byrne, malcolm, dempsey, carr, farrell, hart, sinclair
486	nations, africa, european, african, europe, countries, hemisphere, asian, global, asia
945	lewsey, carling, cueto, tindall, olly, balshaw, guscott, monye, catt

Table C.9: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `rugby` class of BBC Sport dataset

Axis	Members
314	roddick, federer, nadal, henman, hewitt, agassi, safin, rusedski, lleyton, marat
424	serena, clijsters, venus, hardenne, henin, capriati, hingis, justine, navratilova, martina
37	moya, ferrero, nalbandian, robredo, coria, gaudio, conchita, chela, gaston, feliciano
999	seed, seeds
435	slam, wimbledon, singles, doubles, clay, roland, semis, garros
707	set, open, break, opening, ended, broke, sets, breaks, closing, closed
924	kuznetsova, sharapova, dementieva, svetlana, petrova, davydenko, youzhny, nikolay, bondarenko, zvonareva
738	mauresmo, dechy, rochus, nathalie, amelie, santoro, fabrice, monfils, grosjean, razzano
1012	davenport, lindsay, marion, haas, becker, pierce
223	ljubicic, jelena, ancic, srichaphan, ivanovic, jankovic, dokic, paradorn, sprem, karolina

Table C.10: Top 10 word clusters and their members when sorted based on the decreasing order of sum of weights for `tennis` class of BBC Sport dataset

#	Terms (Associated Weight)
1	said (0.22), mr (0.16), year (0.13), would (0.11), game (0.11), us (0.1)
2	mr (0.27), labour (0.2), elect (0.2), blair (0.16), parti (0.16), tori (0.14), tax (0.13), brown (0.13), govern (0.13), minist (0.11)
3	labour (0.17), elect (0.16), blair (0.15), mr (0.14), parti (0.14), win (0.12), tori (0.12), england (0.11), brown (0.11), play (0.1)
4	film (0.49), award (0.28), best (0.21), star (0.16), oscar (0.16), nomin (0.15), actor (0.14), actress (0.11), festiv (0.1)
5	mobil (0.22), phone (0.2), use (0.17), peopl (0.15), technolog (0.15), user (0.13), game (0.13), comput (0.12), music (0.11), servic (0.11), softwar (0.1)
6	yuko (0.18), court (0.14), compani (0.14), firm (0.13), law (0.13)
7	music (0.3), band (0.25), album (0.23), chart (0.18), song (0.15), record (0.14), singl (0.14), top (0.12), olymp (0.11), number (0.11), rock (0.1)
8	game (0.31), club (0.17), music (0.14), unit (0.13), sale (0.13), mr (0.12), yuko (0.11), player (0.11), share (0.1)
9	england (0.26), music (0.19), wale (0.19), band (0.17), album (0.14), ireland (0.14), robinson (0.12), rugbi (0.12), chart (0.1)
10	mobil (0.4), phone (0.32), club (0.11)
11	yuko (0.26), game (0.19), mobil (0.17), wale (0.14), ireland (0.13), russian (0.13), england (0.13), court (0.12), franc (0.12), phone (0.11), oil (0.11), tax (0.1)
12	game (0.32), lord (0.18), athlet (0.11), eu (0.1)
13	yuko (0.22), lord (0.16), seed (0.14), govern (0.13), law (0.12), play (0.12), russian (0.11), oil (0.11), court (0.11)
14	award (0.37), best (0.31), tax (0.19), game (0.18), yuko (0.16), nomin (0.15)
15	eu (0.17), airlin (0.13), countri (0.12), plan (0.1), european (0.1)
16	mr (0.24), blair (0.22), brown (0.19), award (0.19), eu (0.17), best (0.16), minist (0.14), game (0.13), prime (0.11), countri (0.11), mobil (0.11)
17	yuko (0.23), brown (0.22), oil (0.16), music (0.15), film (0.12), aid (0.11), countri (0.11), world (0.11), tax (0.1), russian (0.1)
18	tv (0.22), show (0.19), broadband (0.14), mr (0.12), search (0.1)
19	game (0.21), broadband (0.2), bt (0.14), servic (0.12), brown (0.11)
20	oil (0.16), yuko (0.16), indoor (0.13), race (0.12), broadband (0.11), chart (0.11), minut (0.11), european (0.11), eu (0.1), china (0.1)

Table C.11: Top 20 axes of Latent Semantic Analysis on BBC Dataset sorted and selected in the decreasing order of the singular values. The table also lists the terms (stemmed) weighing at least 0.1 in their association with each of the top 20 axes.

#	Terms (Associated Weight)
1	said (0.16), england (0.16), first (0.12), game (0.11), year (0.11), win (0.1), would (0.1)
2	england (0.19), pakistan (0.17), test (0.16), cricket (0.15), india (0.14), series (0.14), south (0.13), australia (0.13), africa (0.12), day (0.11), tour (0.11)
3	champion (0.18), open (0.17), olympic (0.15), seed (0.15), world (0.14), indoor (0.13), race (0.11), set (0.11), australian (0.11), year (0.11), holmes (0.1), athens (0.1)
4	wales (0.25), ireland (0.23), england (0.19), france (0.18), nations (0.15), robinson (0.15), rugby (0.12), italy (0.12), scotland (0.11), williams (0.11), half (0.11)
5	kenteris (0.2), iaaf (0.18), greek (0.18), thanou (0.17), athens (0.15), drugs (0.14), athletics (0.12), olympic (0.11), olympics (0.11), england (0.11), doping (0.11)
6	indoor (0.16), race (0.14), minutes (0.13), holmes (0.12), european (0.12), record (0.11), ball (0.11), goal (0.11), 60m (0.1)
7	kenteris (0.23), greek (0.21), thanou (0.2), iaaf (0.2), drugs (0.14), minutes (0.11), ball (0.11), doping (0.11), goal (0.11), charges (0.1)
8	south (0.23), england (0.22), africa (0.21), vaughan (0.19), strauss (0.14), flintoff (0.13), trescothick (0.13), boje (0.12), andrew (0.12), ntini (0.12), jones (0.12), smith (0.11), pollock (0.11), harmison (0.1), steyn (0.1)
9	liverpool (0.46), gerrard (0.33), benitez (0.26), steven (0.16), anfield (0.13), morientes (0.12), wales (0.11)
10	wales (0.37), williams (0.27), jones (0.21), ruddock (0.13), thomas (0.13), henson (0.1), scotland (0.1), italy (0.1)
11	chelsea (0.29), mourinho (0.21), england (0.18), robinson (0.15), injury (0.14), bath (0.11), pakistan (0.11)
12	chelsea (0.35), mourinho (0.28), zealand (0.19), sri (0.16), cricket (0.14), lions (0.12), barcelona (0.11), new (0.1), lanka (0.1)
13	zealand (0.17), rugby (0.17), new (0.16), lions (0.14), club (0.12), australia (0.12), liverpool (0.12), newcastle (0.11), injury (0.11), bath (0.1)
14	lions (0.17), rugby (0.16), zealand (0.14), roddick (0.12), australia (0.12), arsenal (0.12), ponting (0.11), players (0.11)
15	united (0.22), cup (0.2), ireland (0.14), manchester (0.12), tie (0.1)
16	radcliffe (0.21), marathon (0.21), race (0.17), ireland (0.16), london (0.15), country (0.13), cross (0.13), paula (0.12), cup (0.12)
17	club (0.18), chelsea (0.15), ireland (0.15), gara (0.14)
18	real (0.34), madrid (0.22), beckham (0.15), owen (0.12), barcelona (0.12), england (0.12), arsenal (0.12)
19	roddick (0.3), seed (0.15), wales (0.13), agassi (0.11), moya (0.11), radcliffe (0.1)
20	united (0.2), robinson (0.13), england (0.12), fa (0.11), ferguson (0.11), football (0.11), manchester (0.1)

Table C.12: Top 20 axes of Latent Semantic Analysis on BBC Sport Dataset sorted and selected in the decreasing order of the singular values. The table also lists the terms weighing at least 0.1 in their association with each of the top 20 axes.

Class	**LSA**	**WcDe**
business	samsung, suppos, procedur, exceed, assum	market, sales, prices, demand, consumer
	game, film, play, award, best	bank, financial, investment, credit, banks
	olymp, athlet, user, mail, viru	economy, economic, global, asia, asian
	mobil, economi, elect, growth, rate	shares, share, stock, exchange, trading
	club, chelsea, arsen, liverpool, leagu	growth, increase, increased, increasing, output
entertainment	samsung, suppos, procedur, exceed, assum	film, films, series, movie, hollywood
	game, england, player, match, club	awards, award, prize, honour, awarded
	film, england, game, wale, oscar	band, rock, pop, hip, hop
	mobil, olymp, phone, champion, seed	actor, actress, starring, starred
	music, eu, european, softwar, appl	music, artists, musical, artist, musicians
politics	samsung, suppos, procedur, exceed, assum	election, parliament, elections, parliamentary, assembly
	game, film, play, award, best	party, leader, political, opposition, politics
	mobil, firm, market, compani, phone	mr, dr, mrs, prof, baroness
	game, england, player, match, club	blair, downing, gates, colin, pentagon
	yuko, tax, phone, firm, mobil	labour, union, association, unions, federation
sport	samsung, suppos, procedur, exceed, assum	game, games, players, team, player
	mobil, firm, market, compani, phone	england, wales, ireland, scotland, irish
	mobil, economi, elect, growth, rate	chelsea, liverpool, manchester, newcastle, leicester
	game, microsoft, softwar, viru, comput	final, match, round, champions, matches
	lord, parti, game, award, profit	roddick, hewitt, nadal, federer, henman
tech	samsung, suppos, procedur, exceed, assum	digital, software, computer, devices, device
	economi, growth, bank, rate, econom	users, online, internet, content, web
	film, seed, mr, festiv, blair	mobile, broadband, wireless, providers, subscribers
	tax, parti, film, tori, yuko	microsoft, apple, google, yahoo, ebay
	india, site, mobil, search, blog	nintendo, xbox, console, ds, consoles

Table C.13: A comparison between the top 5 topics/clusters (represented by 5 terms/words each) of both LSA and WcDe vector spaces. The top 5 topics/clusters were selected on the basis of sum of weights across the document vectors of each class of BBC dataset.

Class	LSA	WcDe
athletics	chelsea, england, arsenal, league, liverpool	olympic, championships, event, competition, athletics
	christos, drew, substances, clever, faking	60m, 200m, 100m, 1500m, 800m
	seed, open, roddick, federer, beat	kenteris, thanou, katerina, annus
	open, said, seed, cricket, rugby	race, grand, finish, finished, runners
	williams, rugby, bangladesh, fifa, zimbabwe	gold, medal, silver, medals, bronze
cricket	christos, drew, substances, clever, faking	pakistan, india, sri, bangladesh, lanka
	chelsea, arsenal, liverpool, league, united	wicket, overs, wickets, batsman, bowler
	robinson, club, pakistan, souness, united	vaughan, trescothick, giles, gillespie, thorpe
	lions, arsenal, souness, wenger, india	ponting, martyn, lehmann, hayden, warne
	seed, lions, rugby, pakistan, woodward	pietersen, flintoff, harmison, hoggard, lbw
football	christos, drew, substances, clever, faking	chelsea, liverpool, manchester, newcastle, everton
	open, said, seed, cricket, rugby	arsenal, wenger, arsene, gunners
	cricket, indoor, rugby, new, world	gerrard, ferguson, hodgson, owen, wilkinson
	arsenal, marathon, radcliffe, jones, chelsea	mourinho, benitez, aragones, rafa, houllier
	england, robinson, ireland, liverpool, wilkinson	souness, redknapp, keegan, moyes, eriksson
rugby	christos, drew, substances, clever, faking	england, ireland, wales, scotland, irish
	pakistan, cricket, india, test, chelsea	wasps, dallaglio, saracens, corry, vickery
	chelsea, arsenal, liverpool, league, united	jones, williams, robinson, smith, johnson
	seed, open, roddick, federer, beat	rugby, cricket, fifa, icc, uefa
	pakistan, ireland, cricket, india, team	lions, rangers, saints, wolves, dragons
tennis	christos, drew, substances, clever, faking	roddick, federer, nadal, henman, hewitt
	chelsea, england, arsenal, league, liverpool	serena, clijsters, venus, hardenne, henin
	cricket, indoor, rugby, new, world	moya, ferrero, nalbandian, robredo, coria
	pakistan, india, australia, khan, ireland	seed, seeds
	pakistan, cricket, india, test, chelsea	slam, wimbledon, singles, doubles, clay

Table C.14: A comparison between the top 5 topics/clusters (represented by 5 terms/words each) of both LSA and WcDe vector spaces. The top 5 topics/clusters were selected on the basis of sum of weights across the document vectors of each class of BBC Sport dataset

Appendix D

Implementation Details

This chapter gives details of the implementation of different methodologies compared in this thesis. All the experiments were implemented in Python. There are majorly 3 python libraries that implement many of the methods discussed in this thesis - `scikit-learn`, `gensim` and `nltk`. In this chapter, I'll provide details of which methods from these libraries were used to implement different representation techniques evaluated in Chapter 4. Since these libraries are often updated, the definitions of the methods may change over time. Therefore, the versions of these libraries used at the time of implementation, are given in Table D.1.

Python package	**Version**
`scikit-learn`	`0.23.2`
`gensim`	`3.8.0`
`nltk`	`3.5`

Table D.1: Relevant python libraries and their versions

Pre-Processing

The pre-processing of documents was implemented using methods from `nltk` library -

`RegexpTokenizer` from `nltk.tokenize` package was used to split the documents into tokens based on regular expression. For the sake of simplicity and consistency in tokenization, the regular expression used to tokenize the documents is the same as the default regular expression defined for `sklearn.feature_extraction.text.CountVectorizer`[1], i.e. `(?u)\b\w\w+\b`.

`nltk.corpus.stopwords` provides stopwords for various languages. This provides the list of stopwords for the English language. Wherever the stopwords were removed during the pre-processing, this list of stopwords was used.

[1]Token pattern defined in the github repository of `scikit-learn` - link to the line of code

`PorterStemmer` from `nltk.stem.porter` package is used for stemming the words wherever the words were stemmed during the pre-processing.

Text Representation Methods

The techniques used to represent documents or words were mostly implemented using the `sklearn` and `gensim` libraries. Wherever applicable, the methods, classes, packages, and the libraries used in the implementation of various methods discussed in the thesis are given below -

Document-Term Vector Space Model - `TfidfVectorizer` from `sklearn.feature_extraction.text` was used to generate Document-Term Association Vector.

Latent Semantic Analysis -

- `TfidfVectorizer` from `sklearn.feature_extraction.text` was used to generate TF-iDF Document-Term Association Matrix.
- `TruncatedSVD` from `sklearn.decomposition` was used to decomposing TF-iDF Document-Term Association Matrix using Truncated Singular-Value Decomposition.

Latent Dirichlet Allocation -

- `TfidfVectorizer` from `sklearn.feature_extraction.text` was used to generate Binary Document-Term Association Matrix.
- `LatentDirichletAllocation` from `sklearn.decomposition` was used to generate LDA document representation based on the Binary Document-Term Association Matrix.

Smooth Inverse Frequency weighted average of semantic word vectors was implemented using the GitHub repository published by the authors of Arora et al. (2017). The repository is available at `https://github.com/PrincetonML/SIF`[2].

Word2vec - `Word2Vec` from `gensim.models` was used to train semantic word vectors using Word2vec model.

Doc2vec - `Doc2Vec` from `gensim.models.doc2vec` was used to generate Doc2vec document representations.

Word Cluster based Document Embedding -

- `Word2Vec` from `gensim.models` was used to train semantic word vectors using Word2vec model.
- `AgglomerativeClustering` from `sklearn.cluster` was used to cluster the semantic word vectors using Agglomerative Hierarchical Clustering algorithm.
- `KMeans` from `sklearn.cluster` was used to cluster the semantic word vectors using K-Means clustering algorithm.

[2]Accessed on 2021/08/07

Document Clustering

The document vectors were clustering using the implementation of K-Means clustering algorithm available in `KMeans` class from `sklearn.cluster` module.

Term Index