A Model for Recommending Related Research Papers: A Natural Language Processing Approach

by

Juandre Anton van Heerden

A Model for Recommending Related Research Papers: A Natural Language Processing Approach

by

Juandre Anton van Heerden

Dissertation

submitted in fulfilment
of the requirements
for the degree

Master of Information Technology

in the

Faculty of Engineering the Built Environment and Technology

of the

Nelson Mandela University

Supervisor: Prof. Reinhardt A. Botha

Co-supervisor: Prof. Bertram P. Haskins

April 2022

Declaration

I, Juandre Anton van Heerden, hereby declare that:

- The work in this dissertation is my own work.
- All sources used or referred to have been documented and recognised.
- This dissertation has not previously been submitted in full or partial fulfilment of the requirements for an equivalent or higher qualification at any other recognised educational institute.

Juandre Anton van Heerden

This thesis is dedicated to the memory of my father, **Anton van Heerden**.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisors Prof. Reinhardt A. Botha and Prof. Bertram P. Haskins for their continuous support during my Master's study. Furthermore, I would like to express my gratitude for their patience, motivation, enthusiasm, and immense knowledge. Their guidance assisted me throughout the research and writing of this dissertation. I could not have imagined having better supervisors for my Master's study.

Furthermore, I would like to thank the following benefactors for their financial assistance:

- The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the authors, and are not necessarily to be attributed to the National Research Foundation.

- The financial assistance of the Nelson Mandela University's Post Graduate Research Scholarship (PGRS) is also hereby acknowledged.

Finally, I must express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement throughout the process of researching and writing this dissertation. This accomplishment would not have been possible without you. Thank you.

Abstract

The volume of information generated lately has led to information overload, which has impacted researchers' decision-making capabilities. Researchers have access to a variety of digital libraries to retrieve information. Digital libraries often offer access to a number of journal articles and books. Although digital libraries have search mechanisms it still takes much time to find related research papers.

The main aim of this study was to develop a model that uses machine learning techniques to recommend related research papers. The conceptual model was informed by literature on recommender systems in other domains. Furthermore, a literature survey on machine learning techniques helped to identify candidate techniques that could be used.

The model comprises four phases. These phases are completed twice, the first time for learning from the data and the second time when a recommendation is sought. The four phases are: (1) identify and remove stopwords, (2) stemming the data, (3) identify the topics for the model, and (4) measuring similarity between documents.

The model is implemented and demonstrated using a prototype to recommend research papers using a natural language processing approach. The prototype underwent three iterations. The first iteration focused on understanding the problem domain by exploring how recommender systems and related techniques work. The second iteration focused on pre-processing techniques, topic modeling and similarity measures of two probability distributions. The third iteration focused on refining the prototype, and documenting the lessons learned throughout the process. Practical lessons were learned while finalising the model and constructing the prototype. These practical lessons should help to identify opportunities for future research.

Contents

List of Tables

List of Figures

Listings

Chapter 1

Introduction

Academic content generation has seen an increase in recent years. Getting usable research papers has become a problem. In order to produce reputable and quality papers, vast amounts of research need to be consulted in several domains and sub-domains. Reading and working through the additional research papers can be time consuming and could result in researchers overlooking important topics and discussions.

This study is a exploratory study into how machine learning (ML) coupled with natural language processing can aid in identifying usable research papers with less effort. However, it quickly became clear that machine learning and natural language processing is rich in aiding techniques and algorithms.

This chapter sets the groundwork of the research project. The problem area consists of a brief introduction into machine learning, topic modeling and clustering techniques, leading to the problem statement. In addition, research objectives are identified. Thereafter, the research process is described which was followed to achieve the research objectives. The chapter ends with the chapter outline for the rest of the dissertation and a brief conclusion to the chapter.

1.1 Background

Information overload is a real phenomenon in our digital age, and our access to knowledge and resources have exceeded our capacity to comprehend. The emergence of online databases has made the ability to search, find, retrieve and summarise documents increasingly difficult. The possibilities have increased drastically in recent years of using machine learning (ML), information retrieval (IR) and natural language processing (NLP) to ease this immense task of navigating through the same databases.

Burkov (2019) describes machine learning as a subfield of computer science, which uses algorithms to help identify certain objectives. There are three paradigms of machine learning:

1. Reinforcement learning – primarily focuses on training machine learning models to perform a sequence of decisions based on data (Mousavi, Schukat, & Howley, 2018).

2. Supervised learning – is a machine learning task that maps input to output based on tagged or labelled datasets (Singh, 2019).

3. Unsupervised learning – is defined as mapping inputs to outputs but the datasets do not contain any tagging or labels (Hastie, Tibshirani, & Friedman, 2009).

Information retrieval is a mixture of information systems, databases and data mining techniques (Baeza-Yates & Ribeiro-Neto, 1999). Furthermore, Baeza-Yates and Ribeiro-Neto (1999) mentioned that information retrieval includes techniques which aid searching for information in a document. Another technique, called text mining, also intersects information retrieval and machine learning.

The text mining approach seeks to identify words and phrases that could explain certain underlying structure in the data. Text mining has focused on analysing co-occurrence data through association rules, distribution analysis, and different clustering approaches. Utilising these approaches to create practical categories rather than using predefined categories, opens up a world of new research options. Krasnov (2018) suggests that topic modeling would be a useful tool to extract information from textual data.

1.2 Description of Problem Area

As mentioned in the previous section, information overload has become a real problem in recent years. More specifically, researchers have indicated that information overload has caused three main difficulties (Al-Kumaim, Hassan, Shabbir, Almazroi, & Al-Rejal, 2021). They are; difficulty to process the information which was collected, difficulty to find relevant and quality information and lastly, there are just too much information generated.

Researchers experience difficulty when seeking information they develop cognitive barriers (Savolainen, 2015). Five sub-types of cognitive barriers are reported by the study: (1) not able to differentiate between the information needs and the research needs, (2) inability to formulate information needs, (3) not aware of the various information sources, (4) low self-efficacy, and (5) unable to deal with information overload.

The best way to combat information overload from an personal and technological perspective, is better designed information systems. Such information systems will enable the researcher to better balance the scale between consuming information and understanding it (Bawden & Robinson, 2020).

As mentioned in the previous section, information overload has become a real problem in recent years. More specifically, researchers have indicated that information overload has caused three main difficulties (Al-Kumaim et al., 2021). They are: difficulty to process the information which was collected; difficulty to find relevant and quality information; and lastly, there is just too much information generated.

Researchers experiencing difficulty when seeking information develop cognitive barriers (Savolainen, 2015). Five sub-types of cognitive barriers are reported by the study: (1) not being able to differentiate between information needs and research needs, (2) inability to formulate information needs, (3) not being aware of the various information sources, (4) low self-efficacy, and (5) inability to deal with information overload.

The best way to combat information overload from both a personal and a technological perspective, is to have better designed information systems. Such information systems will enable the researcher to balance the scale better between consuming information and understanding it (Bawden & Robinson, 2020).

This leads to the following problem statement:

To identify related research papers is a time consuming cognitive barrier.

1.3 Research Objective

Research objectives are derived from the problem statement. In order to satisfy the problem described above, the primarily research objective is to:

Develop a model to recommend related research papers

This model is to be developed with the intention of aiding academics in any research field. For this to be achieved, the following sub-objectives must be achieved:

SO1: To identify recommender systems techniques and how they are used.

SO2: To identify machine learning techniques that assist with the recommender task.

The model will include recommender system- and machine learning techniques. The development of the prototype is to show that the model is feasible to implement. Furthermore, it serves to demonstrate applicability by using information security South Africa data. Throughout the development of the model and prototype practical lessons were learned, which will be discussed in Chapter 7.

In the next section, the research process will be discussed.

1.4 Research process

Owing to the fact that this research study is primarily experimental in nature, a model and a prototype were developed. A broader explanation of the paradigm, the research position itself, and also which methods were used throughout the study, are presented in Chapter 4.

1.5 Delineation

This study included the development of a prototype that was built using text mining and natural processing techniques. Furthermore, the techniques used were not chosen based on experimentation but were derived from literature. The prototype was developed to identify topics, however the prototype needs a certain level of manual intervention to validate the recommendations.

The prototype was developed in the Python 3 environment and used several NTLK and Gensim libraries. The dataset was obtained from the Information Security South Africa website.

1.6 Ethical Consideration

No ethical clearance was needed to complete this study. All of the data was used from a publicly available source and do not contain personal or sensitive data. No human interaction was needed to complete this study.

1.7 Chapter Outline

Figure 1.1 provides the layout of this dissertation. It can be categorised into three parts: *Introduction, prototype development, and lessons learned and conclusion.*

The *Introduction* part includes Chapter 1 to 3. Chapter 1 provides the introduction and background to this study. This is followed by the problem statement and the objectives. Lastly, it refers to the research process, which will be discussed in Chapter 4. Chapter 2 focuses on recommender systems and what constitutes a recommendation task. Furthermore, it discusses each recommender system method, along with examples of each. Chapter 3 investigates the various machine learning technologies that were considered in this study. Chapter 3 concludes by investigating document clustering and which techniques should be used.

The *model development* part includes Chapters 4 to 6. Chapter 4 focuses on the research process which was followed in this study, highlighting the research methods used. Chapter 5 deals with the development of the model derived from the research design chapter. Chapter 6 outlines the techniques and methods that were used in the development.

As seen in Figure 1.1, Chapter 6 is created with a dual purpose: (1) while developing the prototype, certain parameters were identified which influenced the quality of the prototype and model, (2) using recommender system concepts, machine learning algorithms and data from an information security conference showed that the model was applicable and feasible.

The *lessons learned and conclusion* part includes Chapter 7 and Chapter

8. Chapter 7 focuses on practical lessons learned during the development of the model and prototype. Chapter 8 concludes the study by summarising it and revisits the research objectives and hints to further research to be done.

The *lessons learned and conclusion* part includes Chapter 7 and Chapter 8. Chapter 8 focuses on practical lessons learned during the development of the model and prototype. Chapter 8 concludes the study by summarising it. It revisits the research objectives and hints at further research to be done.

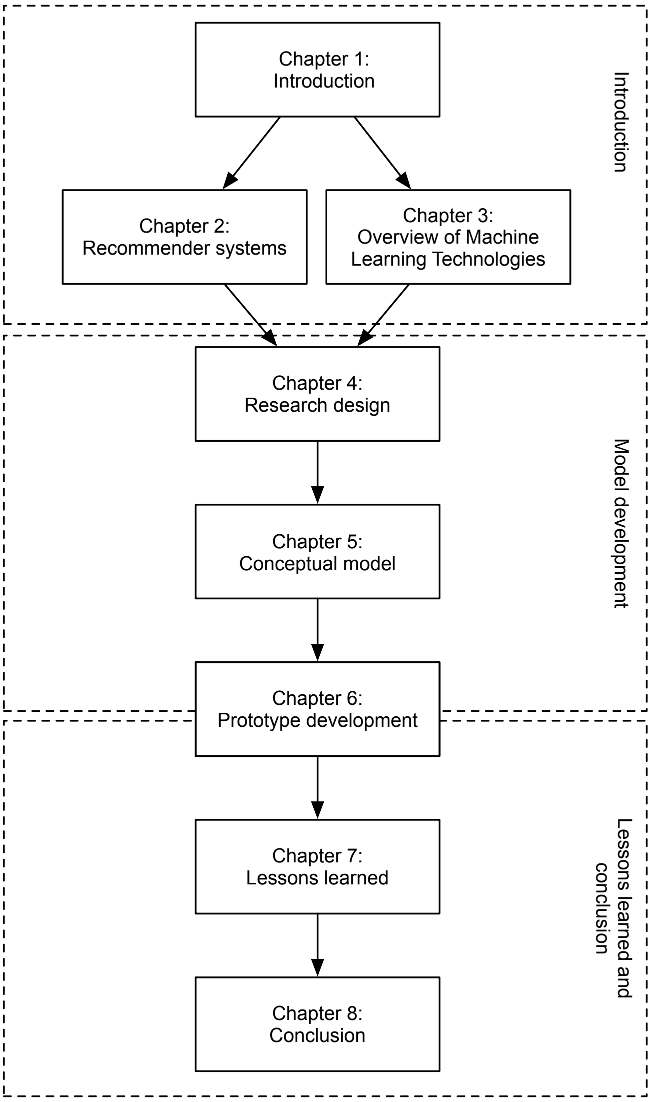


Figure 1.1: Layout of the Dissertation

Chapter 2

Recommender Systems

The focus of this chapter is to identify a research problem and methodology which needs to be followed.

In this chapter, we discuss the background and related work of recommender systems. The following section discusses the history of information retrieval and how it ties in with information filtering. After that, the various methods of recommender systems will be addressed. In the latter sections, a high-level overview will be discussed of the components in a recommender system. The chapter concludes with state-of-the art recommender systems.

2.1 Information Retrieval

We live in an information age surrounded by technology. Information is the lifeblood of the technological ecosystem that is spread using smartphones, tablets, and other computing devices. Owing to technological advances, information is easily created or distributed. It is apparent that with the ever-increasing creation of information, researchers are experiencing information overload, which has a direct influence on academic performance (Suhaimi & Hussin, 2017).

Information overload can be defined as when an individual, who needs to make a decision, is trying to ingest a large amount of data, and the amount of data is larger than the individual's capacity to process the information (Heylighen, 2002). Researchers have learned to combat information overload by employing tools consisting of new technology to expose only the information that is relevant to them, known as information filtering (Hanani, Shapira, &

Shoval, 2001).

Information filtering systems stretch across multiple domains and are useful for extracting information out of unstructured or semi-structured information bodies like e-mails and documents, for large amounts of text, and lastly, it can also keep account of the activities of various user profiles.

The above features are not only limited to information filtering systems but also information retrieval. It is important to remember that information filtering and information retrieval do work similarly, but differ in characteristics indicated below, and in summary in Table 2.1.

1. Frequency of use – information retrieval [IR] systems are designed for one-time, ad hoc users, whereas information filtering [IF] systems are created for repetitive uses.

2. Representation of information needs – in information retrieval systems, users interact with queries. In contrast, in information filtering systems, the long-term needs of users are best saved in user profiles.

3. Goal – information retrieval systems select the information stated in the query out of the database. Information filtering systems just filter out irrelevant data.

4. Database – information retrieval systems usually employ static databases to store and retrieve information. Information filtering works with dynamic data.

5. Types of users – information retrieval systems do not know the users, and anyone can pose a query. Users of information filtering systems need to be known, since the system has models of all the user profiles.

6. Index – information retrieval systems index data based on items and information filtering systems index based on user profiles.

As highlighted in Table 2.1, information retrieval and information filtering does work similarly but there are some differences at their core. Resnick and Varian (1997) mention that sources suggest using recommender systems to tailor content to users.

The aim of information filtering (IF) is to show only the items that are relevant to the users (Hanani et al., 2001). The idea was that Information

Characteristics	**Information Retrieval**	**Information Filtering**
Frequency of use	ad-hoc use	long term users
Representation of information needs	queries	user profiles
Goal	selecting relevant items	filtering out irrelevant data
Database	static	dynamic
Type of users	not known to the system	known to the system, a user model is saved in the system
Index	items	user profiles

Table 2.1: Comparison between IR and IF systems

Filtering can be more effective when humans are involved in the filtering process (Hanani et al., 2001).

Because the spike of product information makes it harder for users to find what they are looking for online, e-commerce sites like Amazon.com use collaborative filtering based on purchase history and customer ratings to make personalised recommendations.

Since the introduction of collaborative filtering in the 1990s, recommender systems have grown to be a very important research area (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994; Shardanand & Maes, 1995). The term 'collaborative filtering' was coined in 1992 by Goldberg when implementing one of the first spam-filtering systems (Goldberg, Nichols, Oki, & Terry, 1992).

'A recommender system is defined as a tool that can recommend a list of items to a particular set of users based on the user's preferences' (Ricci, Rokach, & Shapira, 2011, p. 4).

They have successfully applied recommender systems to different domains such as e-commerce, movies, news, music, research, just to name a few. For example, on Amazon and Netflix users buy and watch more content that is recommended to them (André et al., 2018) now that the gravity of recommender systems is emphasised.

Adomavicius and Tuzhilin (2005) have identified three types of recommendation methods. They are:

1. Collaborative filtering (CF)

2. Content-based filtering (CBF)

3. A hybrid method

In the following section, we will discuss the various types of recommendation methods.

2.2 Collaborative Filtering (CF)

Collaborative filtering systems assume that a user will like the same items as another user liked in the past. Collaborative filtering systems are popular and are commonly used in online shopping websites (Nilashi, Jannach, Ibrahim, Esfahani, & Ahmadi, 2016).

Collaborative filtering focuses recommendations based on similarities between user ratings. Content-based filtering works a bit differently from collaborative filtering, in that users get recommendations based on other users' preferences in the past. Lastly, the hybrid recommender system employs both collaborative filtering and content-based filtering methods.

In recent years, researchers adapted the traditional content-based filtering approach to move towards preference-based filtering. This subdomain focused on predicting items based on the users' preferences (William, Robert, & Singer, 1999; Freund, Iyer, Schapire, & Singer, 2003; Jin, Si, & Zhai, 2002; Jin, Si, Zhai, & Callan, 2003). For example, preference-based filtering can predict movies based on their order relative to each other and not on individual ratings. As stated in the delineation of this dissertation, the focus will be on rating like-based recommendations since it is a popular approach (Park, Kim, Choi, & Kim, 2012).

The collaborative filtering (CF) approach works by predicting user preferences for items through learning from past user-item relationships (Celma & Herrera, 2008). Users give feedback to the system through their preferences or ratings and then the recommender system provides a list based on the feedback.

One of the first recommender systems that used the collaborative filtering method was Ringo, a music recommender system (Shardanand & Maes, 1995). A few other systems that have employed collaborative filtering in that time period can be found in Resnick et al. (1994).

Each method has several advantages and disadvantages, which would guide recommender system implementation to fit the use case. An argument in favour of using collaborative filtering is that implementing Memory based filtering is easy (Bokde, Girase, & Mukhopadhyay, 2015). For example, memory-based methods use historic rating data between users or items to recommend items to people. Another positive aspect is that memory-based filtering would be preferred when new data is continuously supplied to the model (Jannach, Zanker, Felfernig, & Friedrich, 2010). Full updates can be made continuously to the recommending system.

In contrast to the advantages of collaborative filtering, it also does have disadvantages. Some occur when collaborative filtering runs into a general issue called the cold start problem. That is when a new user is introduced to the system, and the system does not know what to recommend to the user.

Collaborative filtering is also less scalable since some systems generate recommendations for billions of user-item pairs. Finally, in some cases, collaborative filtering systems have been found to be manipulated by users promoting their own items (Resnick et al., 1994).

Collaborative filtering is classified into two methods: memory-based- and model-based collaborative filtering (Naak, Hage, & Aimeur, 2009).

2.2.1 Model based collaborative filtering

This creates and builds an offline statistical model based on the user-item pairs seen in the training set. Once the model has been built, it is then applied in an online setting to recommend as intended (Jannach et al., 2010). Several techniques are being used in model-based collaborative filtering, such as probabilistic techniques (Pavlov, Manavoglu, Giles, & Pennock, 2004), and graph- based techniques (Clements, Vries, & Reinders, 2009).

The preferred ones are the latent factor models that reduces the dimensionality of the matrix and uncover latent topics between users and items. Some examples of latent factor models include: singular value decomposition (SVD) for matrix factorisation (Clements et al., 2009) and probabilistic latent semantic analysis (pLSA).

2.2.2 Memory based collaborative filtering

Jannach et al. (2010) describe memory-based collaborative filtering as recommendations that are being made on the entire user-item rating matrix. It computes distance or correlation measures to find user/item similarities. Furthermore, memory-based collaborative filtering looks for either neighboured users for the target user, user-based, or pairs of items that are rated by other users (item-based).

In the next section, the content-based filtering method will be discussed, followed by its advantages and disadvantages.

2.3 Content-based Filtering

The content-based approach to recommendation has its roots in the information retrieval (IR) and information filtering (IF) research field. Recently, much research has been undertaken in the recommendation systems, information retrieval and information filtering fields.

Pushing the boundries in terms of what each field can do, content-based filtering methods analyse a set of features of items that are relevant to the user and link the user profile based on those items. In essence, the method links the users to the items (Lops, De Gemmis, & Semeraro, 2011).

There are a few comparisons that can be made between content-based filtering and IR, outlining the differences and similarities (Belkin & Croft, 1992). The common goal between content-based filtering and IR systems is to select items that are relevant to the users.

For example, suppose the user is looking for a similar or set of similar research papers. In that case, the content-based recommender system will recommend research papers based on the themes found in the current paper; thus, leading us to the advantages and disadvantages of content-based recommender systems.

One of the disadvantages found in collaborative filtering, is a strength in content-based filtering. In contrast to collaborative filtering, the cold-start problem does not apply to content-based filtering since the recommendations are not made based on ratings from other users.

In addition to the non-existent cold-start problem, building a content-based filtering recommender system is straightforward and significant in terms of adding incremental data to the models.

On the other hand, content-based filtering does have difficulty in generating the features of the items. There are limits to the number and types of features that one can generate from items.

In addition to the features, domain knowledge is needed. For example, for research paper recommendations, the system needs to have quite a lot of data regarding each document, including data such as what topics are being discussed, and which branch in that specific topic are being covered. The system needs to distinguish between what the user likes and what not (Lops et al., 2011).

Content-based filtering often suffers from over-specialisation since it recommends the same types of item. For example, when inputting a ransomware research paper into the recommender system, it will only show research papers which also contain ransomware-related topics. This issue is called serendipity. Content-based recommender systems tend to lack novelty awareness (Shah, Salunke, Dongare, & Antala, 2017).

In the next section, the high-level architecture of a recommender system will be discussed.

2.4 High level Architecture of Recommender Systems

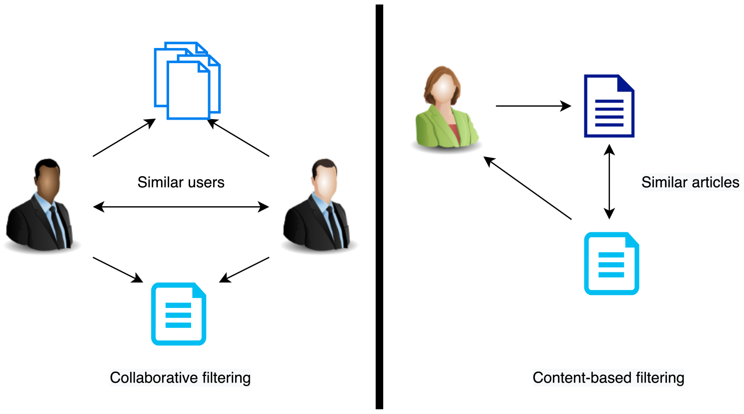


Figure 2.1: Overview of Recommender Systems

This section focuses on introducing the high-level architecture of a recommender system. Each component will be discussed in relation to collaborative filtering and content-based filtering methods.

As seen in Figure 2.1, in both recommender system methods, we see similar components. One component is that of similarity, either between articles or between other users. Both of the methods have a learning component where machine learning or information retrieval methods can be employed.

More components will be introduced later in this section. Recommender systems are strung together to attain a certain goal. The high-level architecture, portrayed in Figure 2.2, will be discussed now:

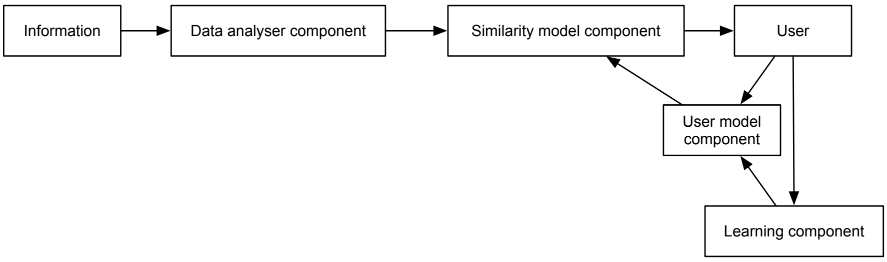


Figure 2.2: Architecture of Recommender Systems

- The data analyser component – the items are obtained or collected (e.g. documents) from information providers. The items are then analysed and represented in a readable format (e.g. in a vector of index terms). Such a vector will be the input to the similarity model component.

- The user/item model component – it gathers information about the user and their needs (explicitly and/or implicitly) and is constructed as user profiles. Then the user profiles will also serve as input to the similarity filtering component. In most collaborative filtering recommender systems, users will have items recommended based on their item model. When the system recommends items, it will compare similar users.

- The similarity model component – it consists of matching the user profile or item models to the corresponding items and calculating the similarity between them.

- The learning component – the user who gets the relevant data item is ultimately the catalyst for feedback. Furthermore, the data is then updated with the new user preference to improve further predictions.

Various methods and techniques can be used to integrate content-based recommender systems: There are many ways to analyse data items and to represent them in better ways, ultimately to gain more knowledge from the user to implement back into the user models. More of those techniques will be discussed in Chapter 3.

Content-based filtering methods can be divided into two categories: (1) models built in machine learning, such as neural networks, naive Bayes model and decision trees (Lops et al., 2011), or (2) heuristic functions stemming from information retrieval techniques (Cantador, Bellogín, & Vallet, 2010; Diederich & Iofciu, 2006). The machine learning (ML) methods try to classify new items that are relevant or not for each user. The authors (Lops et al., 2011) used the naive Bayes model to create a probabilistic model to recommend items to users.

The naive Bayes model calculates the probability whether the item is relevant or not. However, most content-based filtering methods are based on a heuristic approach, which represents users and items are vectors of TF-IDF (Jones, 2004) or BM25 (Baeza-Yates & Ribeiro-Neto, 1999) in a vector space model (VSM).

VSM is a spacial representation of the text documents where each document is represented by a vector in an n-dimensional space. Each dimension corresponds to a term from the overall vocabulary of a document. Recently, the most popular term-weighting scheme is TF-IDF that considers terms occuring frequently in one document, but do not occur as much in the rest of the corpus.

Now to compare the two filtering methods, collaborative filtering and content-based filtering, content-based filtering allows for user independence, which means that it does not use the other user ratings to find the nearest neighbours. Content-based filtering uses the ratings provided by the targeted user to build their own profile. However, one of the most common limitations that occurs in content-based filtering methods is that the content-based filtering methods cannot provide recommendations if the content does not have enough features to distinguish one from another (Lops et al., 2011).

Furthermore, this phenomenon raises the need to create domain knowledge to link new features to new items. For instance, commonly in a movie recommendation system one of the components is to use an external source, like an ontology, to know who the actors and directors are of the movie. Another limitation of using content-based filtering is that it recommends exactly the same content as that which the user already rated. This is called the serendipity problem (Gemmis, Lops, Semeraro, & Musto, 2015). Also, content-based filtering methods cannot recommend items before they have gathered enough data from the user, and this is called the cold-start problem (Lika, Kolomvatsos, & Hadjiefthymiades, 2014).

2.5 Hybrid Method

The term hybrid recommender system comes from the combination of collaborative- and content-based filtering techniques. The combination of the two increases their individual performances by reducing the severity of the cold-start problem. On the other hand, it will diversify the recommendations given to the user. Three base designs of hybrid recommender systems were presented by Burke (2002). These are:

1. One base design is a single recommender system that considers a wide range of input data from other recommendation techniques in one algorithm implementation (Dong et al., 2017). For example, a hybrid system was proposed to combine features like user ratings, and features of the items.

2. Furthermore, another design made an appearance by combining the two methods. The combination was not in series, but rather parallelised (Sharma & Singh, 2016), taking each method's output and recommending both to the user.

3. Lastly, recommender systems are joined together in a pipeline where the output of one recommender system is the input of the other one.

2.6 Summary

The rise of information creation and content on the internet has created a problem. In the problem, it was found that researchers are experiencing cognitive barriers. These barriers include not knowing which pieces of information are relevant to their study, and actually not obtaining the correct information to include in their studies.

Information filtering and information retrieval were discovered, which aided the larger audience to scope down information to only a few lines. Many employed these techniques in various sections in the industry. These advances led to the creation of recommender systems.

Each type of recommender system has specific capabilities. This chapter gave a brief overview of what each recommender system method can achieve. The chapter presented an architectural overview in Section 2.2. The implementation of the recommender system relies on the use case of the implementer.

For the use case in this study, the recommender system would need to have a level of insight with regards to the content. This is necessary as the system would need to identify specific terminology of a given domain; in this case Information Security. This study does not have access to historic recommendation data, but bases its insights on Information Security-related papers.

As described in Section 2.2, collaborative filtering uses item and user pairing to recommend items that other users have liked in the past. Furthermore, collaborative filtering does not look at the content of items. The utility of collaborative filtering did not compliment the use case of the study, and therefore a content based recommender system was used.

The next chapter, Chapter 3, discusses the various technological concepts like machine learning, topic modeling, document clustering, and natural language processing.

Chapter 3

Overview of Machine Learning Technologies

The development in machine learning and information retrieval in recent years has resulted in a surge in recommender systems. However, machine learning and information retrieval plays an integral part in the success and failure of a recommender system. This chapter will cover three popular types of learning: (1) reinforcement learning, (2) supervised learning, and (3) unsupervised learning. This is followed by document cluster, how it works, and challenges pertaining to document clustering. This chapter will also investigate the different problems that natural language processing faces and what tasks are involved. Lastly, topic modeling will be discussed and how it is used.

3.1 Machine Learning

Machine learning (ML) is a subfield of artificial intelligence that is concerned with building algorithms which rely on a collection of some phenomenon (Burkov, 2019). These examples can come from nature, can be created by developers, or can be generated by other ML algorithms. Machine learning is also known as attempting to solve problems by: (1) acquiring a dataset, and (2) automatically building a model using that dataset (Sebastiani, 2002).

The rise of big data in recent years has created the problem of how to translate untouchable data into knowledge. Technological advances have enabled machine learning to solve these problems. Machine learning plays an integral part in the following sectors (Alpaydin, 2010):

- Computational finance – in the credit section of banks, for credit scoring.
- Computer vision – for face recognition, object detection, and motion detection.
- Natural language processing – for text analysis and text summarisation.

Games or simulations were not mentioned in the list above. Simulations bring forth a new need that has to be satisfied. Furthermore, simulation brings a decision-making aspect.

3.1.1 Reinforcement Learning

Reinforcement learning (RL) can be defined as a machine, capable of getting the state of an environment as input also known as features (Mousavi et al., 2018). Actions are executed within each state. Furthermore, each action can give rewards, which can move the machine to another state. As mentioned, reinforcement learning works on a reward system and each component, as indicated in Figure 3.1 will be discussed below.

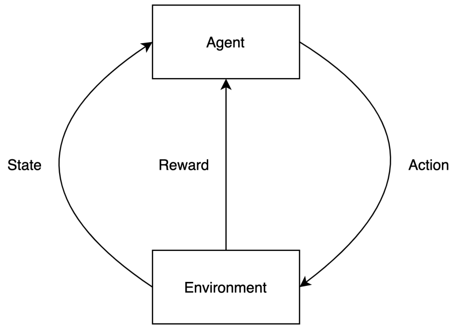


Figure 3.1: Reinforcement Learning reward system (Sebastiani, 2002)

- Autonomous agent - it is the responsibility of the agent to take action.
- Actions - it can be seen as a set of logical steps which are needed to move forward. An action can occur in two states; either reward or penalty.

- Environment - this is the environment in which the autonomous agent finds itself.
- Reward - the main aim of reinforcement learning is to obtain rewards, either good or bad.
- State - this is defined as the position in which it finds itself in the environment. To move around in the environment, the state needs to keep changing.

The goal of a reinforcement learning algorithm is to learn a near-optimal policy that thrives on the reward system. A policy is the rules of the game. Reinforcement learning focuses on addressing problems, which include decision-making (Burkov, 2019). This works well for games, robotics, and logistics.

3.1.2 Supervised Learning

Supervised learning (SL) can be defined as a task that is learning the mapping between input and output by looking at examples of the input and output pairs. It creates a model from the labelled training data consisting of a set of training examples. Labelled suggests that the mapping between questions and answers, or between input and output, has already been done (Singh, 2019). For example, take a financial company that wants to look at users' profiles to decide whether to give them a loan or not. A machine learning model would be trained on historical labelled data, which consists of information regarding the profiles of past customers (Kotsiantis, Zaharakis, & Pintelas, 2007).

The methodology that is used in supervised learning can sometimes vary based on the output of the model. Some supervised machine learning algorithms are listed here:

- Logistic regression
- Decision trees
- Linear regression
- Support vector machines

Another integral part of supervised learning is evaluating the model. Based on the type of model, the appropriate evaluation metric can be chosen and applied (Sebastiani, 2002). Furthermore, this can be done by splitting the training data into two sets: train set, and validation set. Training of the model would be done on the training set and testing the performance should be done on the labelled validation set. Changes can be made in the hyperparameters to improve the performance of the model (Kotsiantis et al., 2007). Hyperparameters are used to configure various aspects of the learning algorithm and they have a direct impact on the results and the performance of the created model.

3.1.3 Unsupervised Learning

In unsupervised learning, a model is trained on similar unlabelled data. Since the data does not contain labels and is sometimes unstructured, the model will just be trained without any influence given by labels. In unsupervised learning, the machine tries to find latent patterns and insights into the data that can be used in any form. The relevant unsupervised learning algorithms are:

- Clustering algorithms (document and hierarchical)
- Dimensionality reduction techniques
- Topic modeling

The rest of Chapter 3 will cover different applications of unsupervised learning, and how they interact with one another.

3.2 Document Clustering

Clustering is the answer to the problem of learning to map a label to examples based on an unlabelled dataset. Owing to the dataset being unlabelled, one must decide whether the learned model is optimal, which makes it much more complicated than supervised learning. Clustering has several use cases ranging from text analysis to anomaly detection. A common use case for businesses is to use machine learning-driven clustering for profiling customers based on their activities and building strategies around these results.

Search engines have been employing clustering by finding similar searches and results in one cluster. Document clustering, a branch of clustering, is the technique that data mining uses, which includes concepts from fields of machine learning, information retrieval and natural language processing. Document clustering organises documents into different groups called clusters, where the documents in the cluster share some common features according to the similarity measure. Clustering, in general, can produce overlapping clusters or non-overlapping clusters. In an overlapping cluster, it is likely that multiple clusters can contain the same document (Andrews & Fox, 2007); in a non-overlapping cluster, the opposite can happen.

An example of supervised learning and unsupervised learning, in terms of document clustering, can be in document classification, where all the classes and their properties are known beforehand. In document clustering, all the properties or other information are unknown. Thus, classification is an example of supervised learning, and clustering is an example of unsupervised learning (Andrews & Fox, 2007). Document clustering can be broken down into two sections: hard clustering and soft clustering (Chen, Tseng, & Liang, 2010). Furthermore, soft clustering can be broken down even more, into partitioning and hierarchical.

- Hard clustering - this clusters the features to exactly one cluster.
- Soft clustering - clusters feature into multiple clusters. For example, if the titles of papers need to be clustered, ‘natural language and information retrieval’ would be clustered in both the cluster names ‘natural language processing’ and ‘information retrieval’.
  - Partitioning - this type of clustering splits the documents into a fixed number of clusters. An example is k-means clustering (Chen et al., 2010).
  - Hierarchical - this is commonly known as taking shape as a tree of clusters.

In this section, we discussed what document clustering is, covering the different types of clustering. In the next section, the applications of document clustering will be discussed.

3.2.1 Document Clustering Applications

As mentioned above, document clustering falls within supervised learning and can be used in various fields like business and science (Jain, 2010). The origin of document clustering research was to improve recall or precision in information retrieval (IR) systems; however, recently the application of document clustering has evolved drastically. The rate of development in technology has spiked research on how to improve document clustering (Alhawarat & Hegazi, 2018; Mekonnen & Abdullayev, 2017).

Document clustering can be applied in different ways (Abualigah, Khader, Al-Betar, & Alomari, 2017): First, it is a system which enables people to find documents that are similar to that which was inputted. Using document clustering enables systems to find other documents that are similar semantically (Shah & Mahajan, 2012). Second, it is useful to organise large numbers of documents into a taxonomic structure. Third, the number of documents in the information ocean created a need to find duplicates.

Clustering use cases includes plagiarism detection, identifying related news stories and fake news, and optimises search engines (Jin, Cao, Zhang, & Luo, 2016). Lastly, in the most basic form, papers are recommended for an academic based on the papers they have already read. This can be done by using clustering and employing other features of the text, ultimately improving the quality of recommendations. The use of latent Dirichlet allocation and content-based filtering is evidence that it can work well, as depicted in Yeh and Wu (2010).

3.2.2 Document Clustering Procedure

Moving from a collection of documents to a cluster of documents a few processes need to be followed. The processes generally comprise three components: (1) feature extraction and selection, (2) document representation and, (3) document clustering (Shah & Mahajan, 2012).

Feature extraction takes the document and applies pre-processing steps to it. Cleaning the text includes the removal of stop words, which should be updated with the domain's most common keywords, which would not add any value from a semantic perspective. The document should then be analysed and features should be extracted (Mugunthadevi, Punitha, Punithavalli, &

Mugunthadevi, 2011). Then, looking at the extracted features, selecting the right ones would be an important exercise to remove further noise. The benefit is that dimensionality is also reduced through only selecting the wanted features; it helps by enabling better data understanding (Wei, Yang, Hsiao, & Cheng, 2006).

After the document has been stripped of all the unnecessary features, the documents are then left with only the features which scored the highest in the metric score (Shah & Mahajan, 2012). Term frequency (TF) would be an example of feature selection metrics. The documents are then grouped into clusters based on their features and the metric scores which were calculated (Wei et al., 2006).

3.2.2.1 Term Frequency-Inverse Document Frequency (TF-IDF)

The dataset that will be clustered will be presented as a set of vectors of an object. Vector space model (VSM) is a model that represents text documents as vectors (Clark, 2015).

Term weight value can be defined as the noteworthiness of a specific term in a document. This can be calculated by the number of times the term occurs within the document over the entire dataset. Term frequency with inverse document frequency (TF-IDF) is the most commonly used term weight scheme (Cui & Potok, 2005). More frequent the words in a document the more important (Peng, Kou, Chen, & Shi, 2006).

3.2.2.2 Dimension Reduction

The increase in vast amounts of data has highlighted the inefficiency of most dimension reduction algorithms (Mugunthadevi et al., 2011). These algorithms are used for feature extraction and feature selection. While feature extraction is taking place, new features are combined with the original features, which causes the computation load to increase. In contrast to feature extraction, feature selection selects the features directly

3.2.3 Similarity Measures for Document Clustering

Cluster similarity is based on the measurements between objects. Three main steps are involved to determine the similarity between objects: (1) Identifiers need to be used to characterise the objects; (2) a weighting scheme needs to be selected; and (3) a similarity coefficient needs to be selected to determine the degree of resemblance between two vectors (Willett, 1988).

Cluster accuracy is the precise distance between a pair of objects must be known in terms of either similarity or distance. Distance and similarity measures have been proposed and are used widely, for example (1) Cosine similarity, (2) Jaccard correlation coefficient, (3) Euclidean distance, and (4) relative entropy (Huang, 2008). An overview of the similarity measures are discussed in Huang (2008).

- Euclidean distance - this is a standard metric for geometrical problems. It is the distance between two points and is the default distance measure used in the k-means algorithm.
- Cosine similarity - the similarity of two documents corresponds to the correlation between the vectors.
- Jaccard coefficient - the Jaccard coefficient contrasts the sum weight of shared terms with the sum weight of terms that are present in either of the two documents but not the shared terms.
- Pearson correlation coefficient - this is an alternative to measure two vectors.
- Kullback–Leibler divergence - this can be used for evaluating the differences between two probability distributions.
- Jensen–Shannon divergence - this is based on the Kullback–Leibler divergence with improved differences, for example, it always returns a finite value and is also symmetric.

Keeping above overview in mind, the Jensen–Shannon divergence has been reported to make a positive contribution when comparing two probability distributions (Uto, Louvigné, Kato, Ishii, & Miyazawa, 2017; Bagul & Barve, 2021).

3.3 Natural Language Processing

Natural language processing (NLP) is a sub-discipline of artificial intelligence and linguistics. The goal of natural language processing was to ease the user's work and to communicate effectively with a computer (Khurana, Koli, Khatter, & Singh, 2017). Since some users are not proficient in machine programming languages, NLP alleviates the pressure that time or lacking the ability to perfect machine language has on a person (Russell & Norvig, 2016).

A language can be defined as a set of rules or a set of symbols (Santana, 2016). The symbols can be combined and used to convey information in a clear and concise manner. This being said, NLP can be classified into two sections: (1) Natural language understanding; and (2) Natural language generation, which means to understand and to generate text as seen in Figure 3.2.

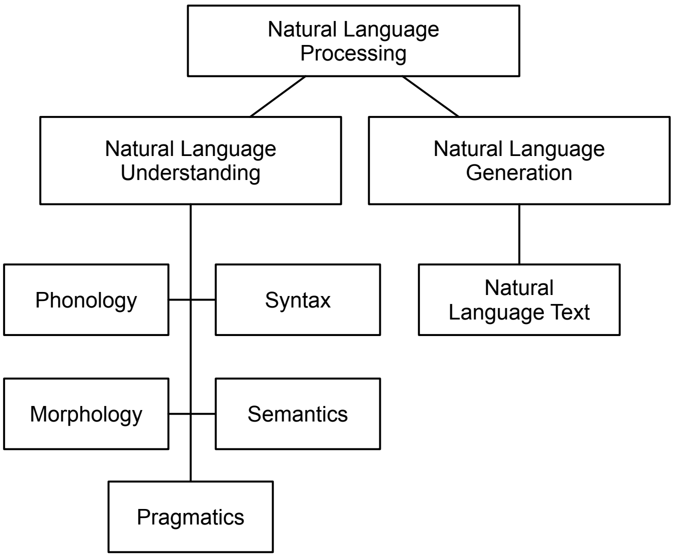

Natural Language Processing
Natural Language Understanding
Natural Language Generation
Phonology
Syntax
Natural Language Text
Morphology
Semantics
Pragmatics

Figure 3.2: Classification of NLP

Linguistics is the domain where languages are studied, which involves the meaning of language, and the context in which language finds itself (Bates, 1995). The important terminologies of NLP are: (1) Phonology, that refers to the relationship in sound, (2) Morphology, which refers to word formation,

(3) Syntax, which is the sentence structure, (4) Semantics, which refers to the arrangement of words and their meaning (Hassan, Tahir, & Ali, 2021) and (5) Pragmatics, which refers to understanding.

To simplify the complexity of natural language processing, it can be broken up into four distinct stages (See Figure 3.3). In a real-world scenario, these stages seldom occur separately. In the overview that follows, it is assumed that the syntactic analysis and semantic analysis is done by the preprocessing. The rest of this section contains the processes shown in Figure 3.3.

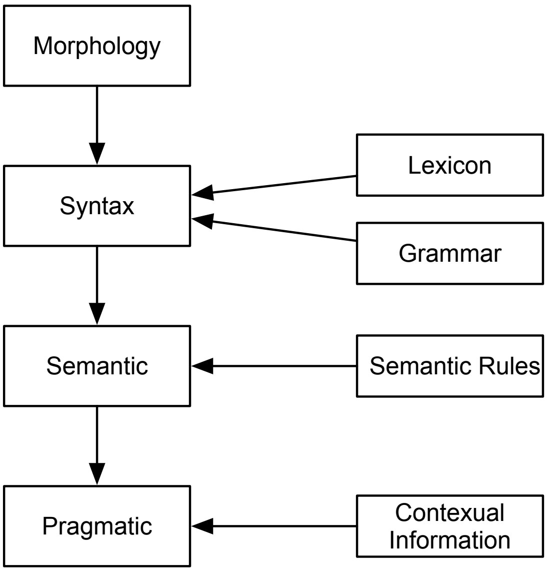


Figure 3.3: Steps in Natural Language Processing

3.3.1 Morphological processing

The first logical step in a typical NLP system is morphological processing. In this step the text will be broken down into sets of tokens corresponding to the equivalent words, sub-words, and punctuation forms (Bates, 1995). For example, a word like 'unnecessarily' can be broken down into three sub-word tokens: un - necessari - ly.

Morphology can be defined as a study of how words can be modified to have similar meanings but used in different syntactical ways. Modifying these words is typically done by adding prefixes or suffixes. Generally, word modification can be broken down into three components:

- Inflection - words can be represented differently based on the syntax in which they find themselves.
- Derivation - new words are made from existing words. ‘Determines’, ‘determining’, and ‘determined’ all come from the root ‘determine’.
- Compounding: new words are made through the grouping of existing words. It is not used so much in English (for example, ‘toothpaste’) but is widely used in other languages.

Outputs from the morphology phase is a set of tokens. These tokens can contain identifiable data that is needed for the parser to do its job. The next stage of processing is syntax and semantic analysis.

3.3.2 Syntax and Semantic analysis

A language processor has certain tasks that it needs to perform: that is, syntax analysis and semantic analysis. There are two main aims for syntax analysis: (1) to check whether a sentence is well formed, and (2) to break up the structure to show syntactic relationships between the words. A syntactic analyser (parser) does this by using a dictionary of words (lexicon) and a set of syntax rules (grammar). The usage of a dictionary and syntax rules indicates how syntactic categories can be combined to form phrases of different types (Nation, Snowling, & Clarke, 2007; Feldman, 1999).

This syntax–semantic combination could deconstruct the sentence ‘The large cat chased the rat’ as follows:

One of the tasks of a language processor is to analyse a sentence and to produce a formal notation that expresses the semantics of a sentence concisely; it is called semantic analysis. When constructing a model, semantic analysis plays the role of finding the meaning of the words in the sentence. In order for that to happen, the dictionary of the model should include whether the words are nouns, verbs or adjectives. The grammar rule in Figure 3.4 with VerbPhrase to Verb, NounPhrase states how the syntactic group is formed.

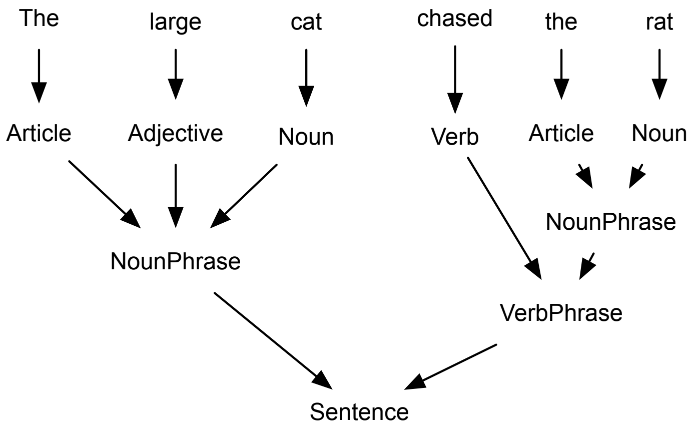


Figure 3.4: Deconstruction of a sentence

The key part of this section is understanding that the syntax and semantic analysis phase is of vital importance for any NLP system or tool. The next stage of processing is semantics and pragmatics.

3.3.3 Semantics and Pragmatic

After the combined stages, syntax and semantic analysis, the next stage of processing is pragmatics. There is no clear distinction between semantics and pragmatics (Mateas & Stern, 2004), but for the purpose of this study we make the distinction as follows: semantics studies the meaning of the word and their meaning within sentences, whereas pragmatics studies the same word and meaning but within a certain context. Doing semantic analysis on a sentence like ‘The large cat chased the rat’ can only provide a string of text which translates to the large cat (the identity of the cat). Pragmatic analysis, like the example supplied, simply maps the actual objects, which exist in a certain context, to a reference obtained during semantic analysis (Russell & Norvig, 2016).

This section has provided examples of how analysing human languages creates certain challenges within the natural language processing domain. In the next section, we will discuss topic modeling.

3.4 Topic modeling

Language models have recently been used to aid speech recognition and handwriting recognition by showing textual data. Language modelling can be defined as a probability distribution derived from words in an indexed vocabulary (Croft, Metzler, & Strohman, 2010). A traditional generative model of a language can be used either to recognise patterns of strings or to generate documents (Sajjadi, Bachem, Lucic, Bousquet, & Gelly, 2018). The generative model, as illustrated in Figure 3.5, is a finite automaton that generates documents.

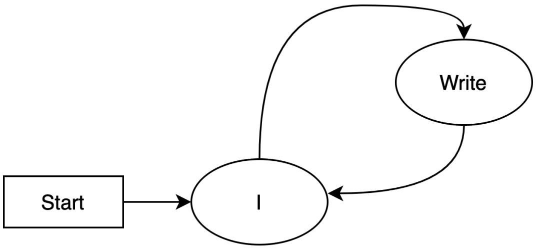


Figure 3.5: A finite automaton and string it generates

Topic modeling has been used as a technique to identify concepts and annotate large text corpora, to keep track of topics over time, and to assess the similarity between topics and documents. The purpose of topic modeling is to analyse data or documents to look for patterns and latent topics. After the topics were identified they would be represented by means of a probability distribution. Topic modeling has been applied actively to several tasks, for the analysis of scientific patterns (Lau, Collier, & Baldwin, 2012; Yi & Allan, 2009; Wei & Croft, 2006; Yi & Allan, 2009) and in scholarly publication search engines (Newman, Noh, Talley, Karimi, & Baldwin, 2010).

Topic modeling refers to a range of generative models for language that specifies procedures by which documents are built (Blei, Ng, & Jordan, 2003). The most preferred algorithm of topic modeling is latent Dirichlet allocation (LDA), which describes a generative model for topics and documents (Blei et al., 2003).

3.4.1 Latent Dirichlet Allocation Algorithm

Latent Dirichlet allocation (LDA) is a type of topic model that associates multiple topics of a document. A topic is a distribution over a fixed vocabulary (Chaney & Blei, 2021). In each topic the distribution of words is different, assuming the topics are specified before the documents are generated. The documents are generated through the following processes. Initially, the random distribution of topics is selected. For each word in the document, a random topic is selected from the distribution of topics. Finally, a word is selected from the topic (Blei et al., 2003; Mekonnen & Abdullayev, 2017).

The goal of topic modeling is to discover topics from a collection of documents automatically. To compute the hidden topic structure from documents, the probability distribution of the hidden variables given must be computed (Mimno, Hoffman, & Blei, 2012).

3.4.2 Topic Model Validation

The quality, performance, and efficiency of the topic model must be evaluated (Ramirez, Brena, Magatti, & Stella, 2012). Topic validations have been created to compare the quality of different algorithms. The first approach is to evaluate the topic models based on perplexity, which is calculated on how well the topics were extracted using the training set and allows the prediction of the occurrence of words belonging to the training set (Ramirez et al., 2012).

Other approaches focus on the semantic coherence of the topics. Chang, Gerrish, Wang, Boyd-Graber, and Blei (2009) introduced human validation of topical coherence via intrusion tests. The judges had to find the intruder in the evaluated topics and if the intruder was easily detected that means the other words have a strong thematic correlation. However, the process requires manual validation of every built model. Automatic approaches have been proposed by Newman, Lau, Grieser, and Baldwin (2010), using point-wise mutual information (PMI) to calculate the co-occurrence in Google search results for all given word pairs in the topic. This approach achieved similar results as human judges (AlSumait, Barbará, Gentle, & Domeniconi, 2009).

3.4.3 Additional Topic Modeling Tools and Techniques

Many researchers have worked on NLP, building tools and systems which made NLP what it is today. There are tools such as sentiment analysers, part-of-speech (POS) taggers, Chunking, name entity recognition, emotion detection, and topic modeling. A sentiment analyser works by extracting sentiments about a given topic. Sentiment analysis consists of a topic feature extraction, sentiment extraction, and association by relationship analysis (Yi, Nasukawa, Bunescu, & Niblack, 2003). Sentiment analysis uses two linguistic resources, namely the sentiment lexicon and the sentiment pattern database (Nasukawa & Yi, 2003). It analyses a document for positive and negative words and gives them a rating on a scale of -5 to +5.

A part-of-speech (POS) tagger can be defined as a piece of software that reads in text and assigns parts of speech to tokens; parts of speech like noun, verb and adjective. POS tagging is a daunting task because a word can represent more than one part of speech at different times. A substantial amount of research has been conducted in European languages, and research has shifted to improve POS taggers for other languages like Arabic, Sanskrit (Tapaswi & Jain, 2012), Hindi (Ranjan, Ray, Harish, Sarkar, & Basu, 2003), etc. It can tag and classify words effectively as nouns, verbs, adjectives, etc. Technological improvements for part-of-speech tagging can work efficiently on European languages but still lacks advancement on Asian languages (Hirschberg & Manning, 2015). The POS tagger used for the Sanskrit language uses the treebank technique (Bengoetxea & Gojenola, 2010). Arabic uses the Support Vector Machine (SVM) (Diab, Hacioglu, & Jurafsky, 2004) approach to tokenise, POS tag, and annotate phrases in Arabic text.

Chunking is also known as shadow parsing, it works by labeling pieces of a sentence with syntactically correlated keywords like noun phrase and verb phrase (NP or VP). Each sentence that is being tagged starts with a unique tag marked as begin chunk (B-NP) tag or inside chunk (I-NP) tag. Chunking can be evaluated by means of the CoNLL 2000 shared task, which provides test data for chunking (Sang & Buchholz, 2000).

The usage of named entity recognition (NER) in places such as the internet is problematic because people do not use traditional or academic English (Nadeau & Sekine, 2007). This brings down the overall performance and quality of language processing tools. By annotating the phrases unlabelled, in-domain and out-of-domain data improves the performance compared to traditional language processing tools (Katiyar & Cardie, 2018).

Emotion detection is similar to sentiment analysis but is used in the social media scene in the mixing of two languages (English and one other language). It categorises statements into six different groups based on emotions, namely sadness, happiness, disgust, fear, surprise, and anger. During the categorising process, the identification of ambiguous words that are in English and the other language should commence by determining the base language of the text. Determining the base language would accelerate the performance and quality of the detection (Khurana et al., 2017).

Event discovery in social media feeds use a graphical model and NER to determine whether it contains the name of a person, city, place, etc. The model operates by listening to noisy data and extracting records and keywords of the events from multiple data streams. Despite the noise and the use of irregular language, the model is able to extract records with very high accuracy (Benson, Haghighi, & Barzilay, 2011).

3.5 Summary

There has been so much achieved in trying to satisfy the problem of information overload over the past few years; hence, machine learning has been used.

Document clustering is a crucial and fundamental pillar in unsupervised document organisation. In this chapter, we have discussed what document clustering is, the types of clustering used, and the applications that employ document clustering. We also discussed the three phases needed to cluster documents, emphasising the importance of extracting and selecting the correct features, which later translate in the quality of clusters. The size of data sets has piqued the interest of researchers to look for alternative ways to reduce the dimensions of an SVM. Furthermore, we have highlighted the types of similarity measures that can be used.

Before using document clustering techniques, the data should first be transformed from raw data to structured data. While working with textual data, to extract meaning out of data natural language processing techniques are commonly used. This chapter discussed three logical steps into to breaking down textual data and removing meaningless words.

Topic modeling was also discussed in this chapter. Topic Modeling is commonly used to identify concepts by utilising document clustering and natural language processing techniques. The chapter introduced the algorithm called latent dirichlet allocation and that perplexity and human validations are used to look at the quality of topics.

The chapter ends by discussing other topic modeling techniques such as sentiment analysers, part-of-speech taggers, chunking, name entity recognition and emotion detection. The next chapter presents the research methodology that was followed.

In the next chapter, the research methodology will be discussed.

Chapter 4

Research Methodology

As mentioned in the previous chapter, machine learning has many technologies that can be harnessed in easing information overload. However, these technologies should be used in a methodical way. This chapter provides the reader with information regarding the overall research design and the research methods used in this study.

The main research approach utilised during this study can be classified as positivism. Research methods that commonly compliment a positivist approach are methods that presents facts and data. Research methods used in this study include: (1) literature review, (2) experimentation, (3) prototype, (4) modeling and, (5) argumentation that threads these methods together.

This chapter provides insights on how the various research methods were implemented in this study, by discussing each in turn.

4.1 Research design

This research study has adopted a positivist approach. The positivist paradigm is based on facts and observation (Wilson, 2014) and reduces the impact that the researcher's interpretation has on the study.

Positivist studies usually come from a deductive research approach, whereas an inductive research approach is manifested within the philosophy of phenomenology (Saunders, Lewis, Thornhill, & Bristow, 2019). The inductive approach can also be known as inductive reasoning, which begins to look for patterns from observations and theories. Moreover, inductive reasoning follows the bottom-up approach, that includes crafting theories or general conclusions from specific observations (Saunders et al., 2019).

For example, when searching for emerging themes in data inductive reasoning is used (Fereday & Muir-Cochrane, 2006). In contrast, deductive reasoning follows the top-down approach, starting with the general and moving to the specific. It uses facts and rules to arrive at its conclusions (Fereday & Muir-Cochrane, 2006).

The scientific research domain makes use of this deductive approach. It enables various theories that are tested. Observational hypotheses can be drawn from them and ultimately be compared to data (Deese & Bechtel, 1990). In the past, scientific research methodologies employed this approach when collecting and evaluating data (Marsden & Pingry, 2018).

In the next section, the methods that were followed in this research study are discussed.

4.2 Methods

This research study was primarily experimental in nature. It did not focus on the researcher's interpretation of the data, but rather on the findings which could be drawn directly from the data. This research study employed a set of methods, which make up the methodology. These methods include a literature review, experimentation, a prototype, and the creation of a model. The upcoming sub-sections provide more detail about how these methods fit into the research objectives and milestones.

4.2.1 Literature review

A literature review is a process of going through various academic studies in search of information dealing with the topic at hand (Olivier, 2009). The literature review in this study focuses on various topics of recommender systems, document clustering, natural language processing and their application to recommend related papers using Information Security South Africa past conference papers. The literature review was conducted to meet secondary research objectives one and two.

4.2.2 Experimentation

As mentioned, the research took on the positivist philosophy. This study was experimental in nature. As Olivier (2009) discussed, an experiment can be conducted with the following goals in mind, namely to test a theory, to prove a theory, and to see whether something interesting happens, which is also known as an exploratory experiment.

Experiments have three main goals; they are (1) to explore a theory, (2) test a theory, and (3) prove a theory. It was later added by Olivier (2009) that it is common for experiments to compare two cases, an older solution to a newer one to see how they compare. In the context of this study, the algorithms and techniques identified were used to construct a prototype to which was used to observe the possibility of using recommender systems, document clustering and natural language processing techniques together to satisfy the primary objective.

Testing can be used to see whether a certain theory holds up against specific cases. In testing a single theory, one should conduct a limited experiment. This may be done to 'feel' whether the theory is correct, to refine the theory even more, or to justify a full-scale experiment. For example, an academic paper recommender system can employ natural language processing techniques to achieve paper recommendations. If it is determined that the experiment holds true, the theory can either be further refined or a full-scale experiment can then begin. In contrast to testing a theory, proving a theory would be conducted to ensure without a doubt that a theory holds true. Proving the previous theory requires all outside factors, which can be a hindrance, to be removed.

Lastly, to see if something interesting happens. This experimental goal has very little structure and provides freedom to play around with certain ideas. For example, an academic paper recommender system can employ natural language processing techniques along with topic modeling algorithms to achieve paper recommendations. Such experiments do not have certain outcomes and are conducted with no given theory. As mentioned in the previous paragraph, the experiments had little structure and were accompanied by a prototype to maximise what there is to learn. The creation of the prototype and experiments were to satisfy the third research sub-objective.

4.2.3 Prototyping

In information technology the term prototype refers to a simplified program or system that serves as an example or demo of the full-scale program or system (Olivier, 2009). A prototype usually only has a few characteristics of the bigger system. The simplicity of the prototype is deliberate because only the study subject matter will be tested or demonstrated. In the research environment, the prototype research method cannot be used solely to constitute the research. In other words, it cannot be used as the icing on the cake; rather, it needs another method to lean on.

Working well with other methods, a prototype can be used in multiple roles. Commonly, there are four roles that a prototype can take: proof of concept, prototype for experimentation, prototype for conceptual clarity, and exploratory research. First, after proposing and constructing a new model or new concept, researchers build a prototype to prove the concept. In other words, the statement can be made that the concept can be implemented and works well in practice. The second role, prototype for experimentation, can be used to gather all the information about the prototype.

In general, information gathered can range from measuring the speed of the system and the quality of a model. However, measuring the speed or quality of a model alone does not make a big research contribution. The third role, prototype for conceptual clarity, is used when a certain concept or work is difficult to visualise. Developing the prototype with this role in mind forces the researcher to focus on the concepts at hand and helps in not overlooking certain details. Furthermore, after the construction of the prototype and its merit can be shown, it can be used as a proof of concept. Lastly, the final role that a prototype can play is in exploratory research.

In all of the other three roles the prototype is constructed to aid the research process. However, when a model, algorithm or concept is not new there are still lessons to be learnt by developing it. For example, in 2003 a new topic modeling algorithm was created called latent dirichlet allocation (LDA) and it achieved great success in the natural language processing (NLP) and information retrieval (IR) domains (Blei et al., 2003). However, there were only a few papers available for the use of LDA in the recommender systems domain. This is a perfect example of incorporating LDA into a recommender system to see what lessons there are to learn. If a major issue is identified while constructing the prototype and it can be linked to the incorporation of LDA and recommender systems, this role of the prototype can be used to create new knowledge. The quality of the research will, however, be determined by the interesting data.

4.2.4 Modeling

A model can be defined to capture the essence of the system or process (Olivier, 2009). In addition, a model needs to be expressed clearly and concisely. In the context of this study, the model was created as to achieve the primary research objective. The construction of the model commenced after the literature reviews, which satisfied sub-objectives one and two. It then became an iterative process between constructing the model, to build the prototype, and to do experiments on it. The feedback from the experimentation ensured amendments to the model.

Furthermore, Olivier (2009) states that a model captures the essence or core of a system or process. All of this while it ignores all the aspects that do not bring value. The model in this study was the main method, and the prototype was created to support it. An experiment can be used to validate a model. For example, defining a model that takes academic research papers that learns trends from the academic research papers. The model can be validated by creating a prototype. Where users can look for recommendations based on one of users papers, which the prototype has never seen. In theory, the model looks like it works; however, using an experiment, the model can be tested in practice (Steenkamp & Mccord, 2007).

4.2.5 Argumentation

Argumentation can be seen as the thread which ties several statements together. As mentioned, arguments string from facts to create a premise on which conclusions can be based. An argument can be supported by other arguments, it can derail other arguments, or highlight main ideas (Walton, 2009).

This study used argumentation throughout the development of the model. Deductive argumentation was used to develop a theory to create and test a high-level model, diving deeper into which algorithms were needed to address the primary objective.

4.3 Summary

This chapter has discussed two major points: (1) the paradigm in which the research finds itself, and (2) the collection of methods which the study employed to achieve the main research objective. The aim of the research was to explore the possibility of using natural language processing and information retrieval techniques to satisfy the research objective; thus making it an exploratory study.

The literature reviews reported on in Chapters 2 and 3 showed viable avenues and options to pursue. This finding ensured that the model would be based on pre-existing knowledge, establishing it in the domain of epistemology. The second half of the chapter covered the different methods used. A survey of the literature to identify trends and algorithms that could be used in the study was captured in Chapter 3. This was followed by creating a prototype of the proposed model and running various experiments with it, adjusting certain values every time.

In the next chapter, the construction of the conceptual model will be discussed.

Chapter 5

Conceptual Model

In Chapter 2, we discussed a brief history and introduced the three main types of recommender systems. The importance of this exercise was to review the literature and to determine the inner workings of how recommender systems work. The corpus of papers revealed that nearly 55% of papers employed the content-based filtering (CBF) techniques of handling the content and the ranking system i.e., user profile building. This led to further exploration to find algorithms and techniques that will complement CBF.

Chapter 3 lay the groundwork and introduces various machine learning (ML) technologies. Primarily, the literature in Chapter 3 was reviewed to shed a light on the field of machine. However, it introduced complex interrelationships within its domain and the interaction with others. These interrelationships created the need to identify and approach employing various technologies covered in Chapters 2 and 3.

In Chapter 4, the approach was not only set out, but various stepping stones also were discovered. The creation of the initial prototype was discussed in this chapter. Furthermore, common traps and concerns were identified, i.e. feature extraction, selecting the number of topics, and the technology used to determine the similarity or to recommend academic papers. In this chapter, the recommendation model, a conceptual model is developed to ease the intricacy surrounding the implementation of a NLP based recommender system.

The recommendation model uses various algorithms and techniques derived from Chapters 2, 3, and 4. In Section 5.1, the model will be discussed from a birds-eye view. Later in the chapter, we will transition from an abstract level to a slightly more technical one. The above-mentioned technical overview will be covered in Section 5.2.

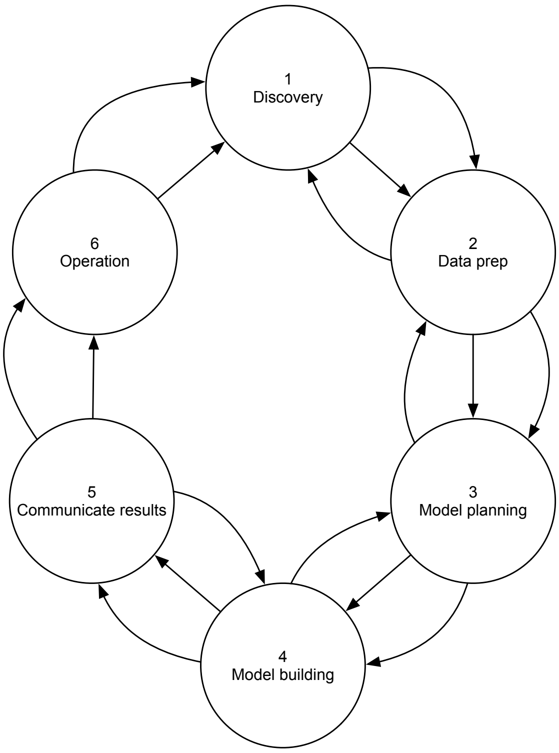


Figure 5.1: Data Analytics lifecycle (Storey & Song, 2017)

5.1 Model Overview

In recent years, research and development have been focused on creating a streamlined, widely acceptable data analytics lifecycle.

The goal of the data analytics lifecycle was to have a structure in place that could aid developers and other researchers. Teams usually learn new things throughout their projects and often need to go back to the previous phase to refine their work based on new insights and information they have uncovered (Dietrich et al., 2015). Each component of the data analytics lifecycle will briefly be discussed below:

1. Discovery – in this phase, discovering and gathering data is undertaken. It is essential to frame the data needs best and to obtain the correct

data according to the requirements.

2. Data preparation – the data must be correctly formatted to be used in a later phase.

3. Model planning – this phase entails looking at various methods, techniques, and workflows that need to be employed to learn about the underlying relationships between the variables.

4. Model building – also known as analyze data, this phase focuses on analysing the data and determining whether the existing tools will suffice to get to the end goal.

5. Communicate results – in this phase, various visualisation techniques will be considered and a summary will be developed to convey to the stakeholders.

6. Operationalise – this is also commonly known as making decisions. This phase focuses on delivering reports, code, and other technical documents.

Known terminology	**Study terms**	**Domain of techniques**
Discovery	Past papers	Dataset
Data prep	Preprocessing	Machine learning - Chapter 4.2.1
Model planning	Learning	ML + IR - Chapter 3
Model building	Human intervention	Prototype - Chapter 4.2.3
Communicate results	Represent in data frame	Discussion - Chapter 7
Operation	Enable decision makers	Evaluation - Chapter 7.5

Table 5.1: Mapping between Terminologies Used in the Study

With the above being said, the researcher felt the need to provide a mapping between the terminology used in the research domain and this study. As seen in Table 5.1, it should be known that when the researcher uses terminology like past papers, pre-processing, and learning they hold the exact content of the corresponding terms used in research. The column on the right in Table 5.1 can be translated to be the domain and/ or section in this study which addressed each phase of this model.

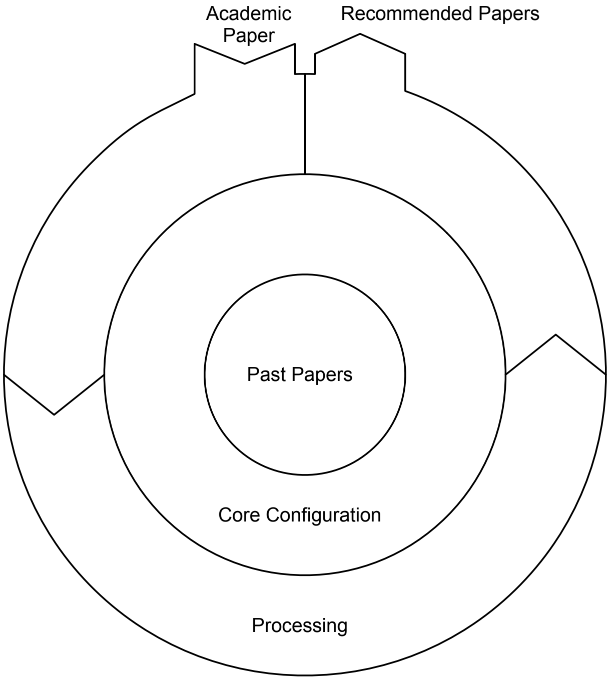


Figure 5.2: The Model Overview

This section of the chapter covers the model overview, as seen in Figure 5.2. The model is categorised into three components: (1) past papers, (2) learning, and (3) processing.

Past papers can be identified as the core of the model. It is made up of past or historical papers. This will then flow into the learning component, which is a fundamental stepping stone in defining what needs to be done. In the processing component, the model looks at the defining functions in learning and further refines them. The learning and the processing component will be discussed further in this chapter.

5.2 Learning

The learning component is created by analysing past papers closely. The goal was to understand better the characteristics which made up the learning component. The conceptual model is constructed in such a way that it is not only information security domain-specific. This being said, the characteristics of the learning component were identified and guided by literature. The characteristics of the learning component are:

1. Populating the stopword list
2. Stemming
3. Topics of the model
4. Measuring the similarities

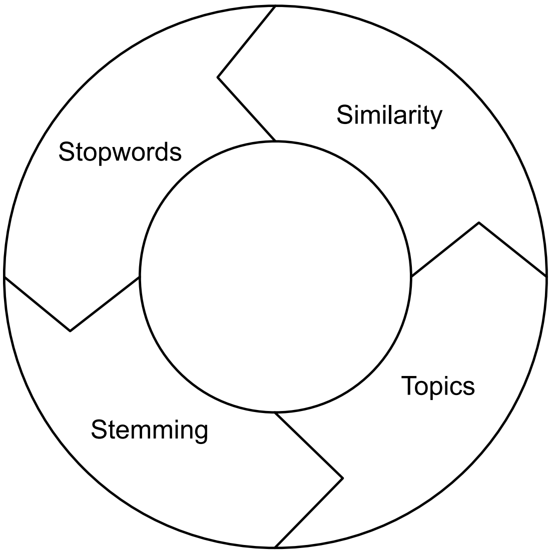


Figure 5.3: Learning Component

The flow of the learning component is using the stopword list, then stemming those words. Stemming leads into topics that will be generated, and the similarity of these topics will be measured. As seen in Figure 5.3, the flow of the learning component leads one character into another. The reason for this is that the learning component is a continuous process. Each of these characteristics is described in the following sections.

5.2.1 Populating stopword list

The stopword list was primarily made up of NLTK's stopword list. The goal of this choice was to employ a stopword list with a comprehensive list of words. NTLK is one of the most widely used NLP libraries. Furthermore, after close consideration, additional words were included in the stopword list as time passed. These words were primarily domain-specific such as: (1) information, (2) technology, and (3) security.

The goal of the inclusion was to retrieve second- or third-tier topics from the text. The textual outputs of each component of the learning component were analysed, and words that did not bring any value or primarily domain-specific words were omitted.

The process of including some words and excluding others was a manual and iterative process. The resulting words were best suitable for the information security domain and were used as the stopword list.

5.2.2 Why Stemming?

Stemming was employed for the sheer simplicity and thorough work it has done. To recap, stemming is removing the suffix of the word to return it to its root form. The Porter stemmer is appropriate to IR research work involving stemming where the experiments need to be repeatable exactly. This being said, after consulting the dataset, it was identified that the domain in which this research position found itself did not need to have a custom-made stemming solution.

After close consideration, the researcher, along with evidence, used Porter stemmer from the NLTK library.

5.2.3 Topics of the model

The information technology domain is such a rich field with regard to topics and sub-topics. It needed to be scoped down to provide better topics for the algorithm to use. For example, information, technology and security would be excluded. As seen in Section 5.3, all of the components fed into one another. In this case, specific topic names were appended in the stopword list.

The result of the methodology mentioned above was smaller topics. These were second- and third-tier topics, such as hacking and man-in-the-middle attacks, respectively. A few of the lower-tier topics fused to make up the second-tier topics. This also holds for second-tier- and first-tier topics.

5.2.4 Measuring the similarities

The vast number of topics made it challenging to group similarities of papers. The topics were too dense from the perspective of dimensionality. For the scope of this research, similarity algorithms reduced dimensionality. In addition to the dimension reduction, the unsupervised learning approach made it viable to use an algorithm to cluster similar topics. This would have a direct impact on measuring the similarities in the academic papers.

5.3 Processing

In this section, the next phase of the conceptual model will be discussed. The academic paper processing phase is executed when a new and unseen academic paper is submitted. Just like the learning component, this phase also has four components. It is merely an extension of the components found in the learning component.

In the first step, the fundamental stepping stone was actioned, the removal of stopwords. The new document had its stopwords removed to simplify and get the text ready for further analysis. Step two includes taking the text that was cleaned and stemming it. This removed the suffix and returned the words to their root form. The goal of Step three was to get the related topics in the text. The last step, Step 4, focused on using the topic output and determining the similarities of the topics. These four components are best described in Figure 5.4, and a brief description of each component is given in the following sub-sections.

5.3.1 Removing the stopwords

The stopwords are removed from the collection of documents. The goal was to remove the words that do not bring any meaning to the sentences. These are terms such as ’specified’ , ’specify’, ’specifying’. The stopword list also included words such as ’information’, ’security’ and ’technology’.

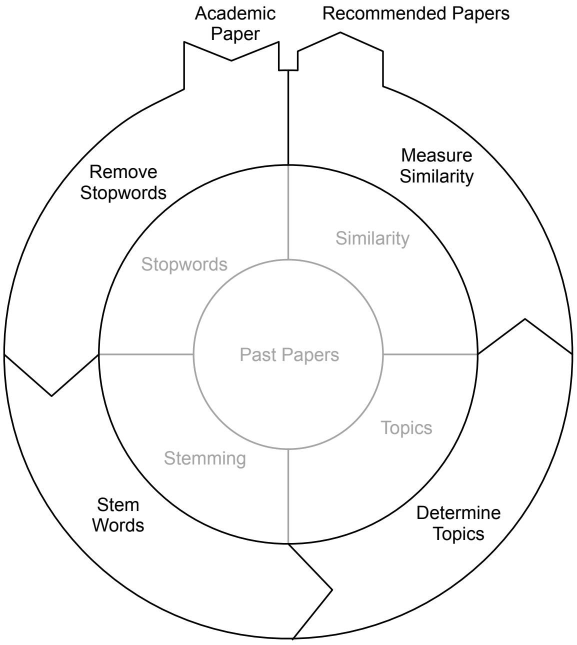


Figure 5.4: Processing phase of the model

5.3.2 Stemming of the words

After the stopwords were removed, the tokenised dataset was then stemmed. For example, the words mentioned above: 'specified' , 'specify', 'specifying' can be stemmed and will look like the following: 'specif', 'specif', 'specif'. As explained in the learning component section of this chapter, stemming is removing the suffix of the word to return the term to its root form. The stemmed words are then used to determine topics.

5.3.3 Determining the topics

This component comprises determining the topics of the dataset. This is done after the stopwords were removed and after the text was stemmed. A topic modeling technique was then applied to the dataset after it was transformed. The soft clustered topics looked something like this: '0.040*"data" + 0.033*"file" + 0.024*"encrypt" + 0.021*"cloud". A similarity ratio of the terms was calculated, and the most similar terms were clustered together, forming topics.

5.3.4 Measuring the similarities

At this point of the conceptual model, the stemmed words that did not contain any stopwords were pushed through the topic modeling technique. The product of the previous three components was a collection of topics that represented the dataset. The collection of topics of the dataset and the topics of the test set were measured in terms of similarity. The lower the score, the more similar the two papers were with one another.

5.3.5 Processing phase summary

In this section, we discussed the academic paper processing phase. This phase was executed when a new document was submitted. The dataset already removes the stopwords, stemmed, and determined the topics. Once a new document was submitted, it went through the same process. The only difference there was that the similarity of the two datasets were then measured. The topics with the lowest score were most similar.

5.4 Summary

The conceptual model presented in this chapter is a template for finding similar academic papers. The use case for this research was focused on information security academic papers. However, it can be used to find any academic papers across disciplines. The conceptual model was built on a collection of academic papers or datasets. Various learning components were identified while surveying the field. The learning components were broken down into four distinct components: (1) stopword, (2) stemming, (3) topics, and (4) similarity.

After the learning component was built through experimentation, it was ready and waiting for new academic papers to do the testing. The academic paper processing phase could then begin. Just like the learning components, the academic paper processing phase also has four components.

First, the dataset was cleaned of all the stopwords. Special consideration was made for the first-tier topics like 'information' and 'security', and those were also included in the stopword list.

Second, once the stopwords were removed, next would be to stem the remaining data. Stemming commonly includes omitting the suffix of a word. This was done to return the word to its root form.

Third, topics were determined by using a topic modeling technique. The technique helped soft cluster all of the similar terms together, which formed topics. These topics consisted of the ’training set’ corpora. Once a ’test set’ document appeared, it was run through the same components.

Once we had the ’training set’ and ’test set’ ready, the last component measured the similarity between the two sets. The new, unseen ’test set’ document was then hard clustered to the most similar set of topics. In the next chapter, the prototype development will be discussed.

Chapter 6

Prototype development

Chapter 5 introduced the conceptual model. This chapter will discuss everything that is needed to develop the prototype and performing the experiments. The prototype essentially provides an implementation of the conceptual model defined in Chapter 5.

The rest of the chapter will look at each element in the pipeline, as depicted in Figure 6.1, especially with the focus on the refinements that were applied. Figure 6.1 illustrates the steps that were followed in the creation of the prototype. Furthermore, Figure 6.1 shows mappings from the prototype development and refinement to the various parts which were created in the conceptual model chapter, in Chapter 5. In addition, the parts that were intentionally neglected will also be discussed. This chapter will start with a overview of the development of the prototype.

6.1 Prototype overview

During the development of the prototype, the focused was to document the process of using one recommender system and discussing the lessons learned from it. The prototype can be split up into four parts at a system level view: (1) identification and removal of the stopwords, (2) stemming the text, (3) topics of the model, and (4) similarity measurement between the test and training set. Although the identification and removal of stopwords and stemming can be grouped under pre-processing, it is important to note that each has an important role in the development of the prototype. Each part plays a critical role in how usable the text is for further topics modeling.

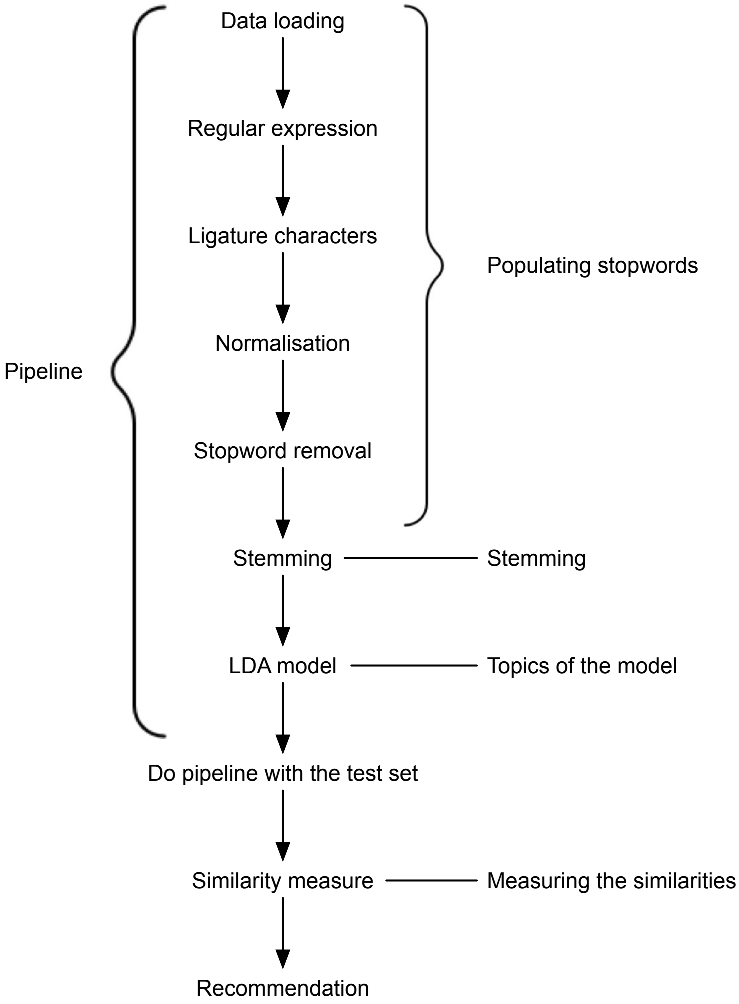


Figure 6.1: High level overview of the prototype

Stopword identification and removal is a process that is applied the same in the learning phase and testing phase of the prototype, the same goes for stemming.

As mentioned in section 4.2.4, it is utmost important to develop a prototype to validate a model. There will always be lesson to be learned, which will be discussed in chapter 7.

The prototype was developed to achieve the following goals:

1. To better understand the problem domain, the research problem, and the solution.

2. To show that the natural language processing and topic modeling approach is feasible.

3. To be used in recommending research papers and to see the relevance of those recommendations made from the prototype.

The prototype has undergone three iterations, which have supported the goals stated above. The first iteration tied in with the first goal, to better understand the domain and to explore a possible solution. The first iteration provided insight by exploring how recommender systems work and the techniques employed to achieve the desired goal. The second iteration focused more on the initial pre-processing phase of the prototype along with the topic modeling algorithm, latent dirichlet allocation (LDA), and what is to be learned from the development. The topics were later analysed to assess whether this approach will be sufficient. Lastly, the third iteration of the prototype was more polished and had refinement applied to it. The focus shifted slightly from NLP and topic modeling to better the quality of the recommendations made by the prototype. Throughout the third iteration, several parameters in the LDA algorithm were changed to better the recommendations. Unless specified otherwise, reference to the prototype will imply the usage of the third iteration of the prototype.

The researcher decided on Python as the base development environment. Python is a programming language that has gained its popularity in recent years for its modularity and effectiveness of handling data. Many developers have built various libraries in aiding the machine learning and information retrieval community with better system building.

More specifically, the researcher used Jupyter Notebook to build and test the prototype. Jupyter Notebook is a web application, which provides a Python development environment for users. Jupyter Notebook can be used in various ways: data cleaning, data transformation, statistical modelling, data visualisation and machine learning. It allows you to create and share documents that contain live coding, visualisations, and text, thus making it perfect for reproducability of the prototype and making the experimentation easier. Every part of the prototype will now be discussed along with the various components related to each part.

Code snippets were used from:

https://github.com/JuandrevanHeerden/Juandre-M-Final

6.2 Identification and removal of stopwords

More frequently than not, looking for papers and analysing them is a daunting task for most researchers. The problem is that those papers found by researchers have too much noise to do the necessary, thus creating the need to clear all the unwanted noise from the text. One of the processes within pre-processing is called stopword removal.

Pre-processing contains many techniques to clear the text and make it ready for analysis. These techniques are primarily used in text mining pipelines to better the text input into Information retrieval systems. These techniques include tokenisation, stopword removal, and stemming and transforming the data into a vector space later in various text mining algorithms.

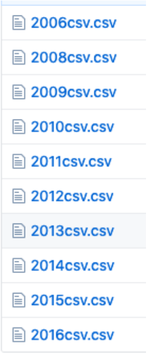
Before stopwords can be removed, they first need to be identified. Each document that needs to be pre-processed contains domain knowledge that is not shared across other domains. Specific domain knowledge needs to be identified first, and then certain words need to be applied to the stopword list. First the data needs to be extracted and analysed. In the next sub-section, we will discuss how the data was extracted. After that, we will discuss the steps taken to identify domain-specific words and add them to the stopword list.

6.2.1 Extracting the data

The data used in this study was obtained from the International Information Security South Africa Conference website. The articles were downloaded and documented in a per year fashion. The abstracts, along with the title of each article, were manually extracted into the form of ten csv files as seen in Figure ??6.2 Each csv file had two columns named title and abstract.

The collection of csv file were uploaded to github for documentation and reproducible purposes. A Python library called Pandas was used to handle the data, as seen in Figure 6.1. Pandas is a well known library, which is primarily used to manipulate data.

```
# Import Dataset
```



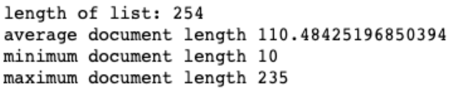
2006csv.csv
2008csv.csv
2009csv.csv
2010csv.csv
2011csv.csv
2012csv.csv
2013csv.csv
2014csv.csv
2015csv.csv
2016csv.csv

Figure 6.2: Data on Github

```
df = pd.read_csv(’https://raw.githubusercontent.com/brix-mix/Fake/master/2006csv.csv’, usecols = [’abstracts’,’title’])
df1 = pd.read_csv(’https://raw.githubusercontent.com/brix-mix/Fake/master/2008csv.csv’, usecols = [’abstracts’,’title’])
frames = [df, df1]
df = pd.concat(frames, ignore_index=True)
df.head()
```

Listing 6.1: Importing data to the system

The dataset consists of 254 documents spread over 10 years. As shown in Figure 6.3, the largest document has 235 words, whereas the document with the least number of words has 10. The average number of words in the documents are 110. What this means is that there are about 20 per cent of the words that will actually have meaning. Usually, out of 100 words used in a piece of writing, 20 have meaning; the rest are filler words or stopwords.



```
length of list: 254
average document length 110.48425196850394
minimum document length 10
maximum document length 235
```

Figure 6.3: Statistics of the documents

It is best practice to remove all the stopwords and unwanted characters from your dataset. Junk in equals junk out. In order to achieve good results, a decision was made to clear the text of unwanted characters. The next sub-section includes further removal of unwanted text.

6.2.2 Regular expressions

Regular expressions can be seen as a sequence of characters that can be used to search for patterns, and those patterns are used to find and replace those characters. The nature of the domain of the document suggests that email addresses and other unconventional characters could be found in the text, therefore Figure 6.2 was used to remove these characters.

```
text = re.sub("((\S+)?(http(s)?)(\S+))|((\S+)?(www)(\S+))|((\S+)?(\@)(\S+)?)", " ", text)
```

Listing 6.2: Regular Expression

This code segment was only run once because all data were already in one big data frame. Looking at the raw text, the researcher found ligature characters, which also needed to be removed and replaced. In the next subsection more details are discussed.

6.2.3 Ligature Characters

In writing or typography, two or more characters can be merged into one to form; a glyph or single character called a ligature. Ligature characters were commonly used in TEX. TEX is a formatting system that was created and made famous in academia. The ease of executing specific typesetting tasks gave TEX its popularity.

This research dataset contains older papers published from 2006 onwards, making it ligature prone. The two ligatures that were found in this research were; fi and fl. The main reason to replace these ligatures is to normalise the text, so that readers who do not change their font encodings will have no problem reading it. As seen in Listing 6.3, the use of regular expressions was employed to find and replace the ligatures, fi and fl, respectively.

```
    # Remove fi and fl ligature characters
    text = re.sub("fi", "fi", text)
    text = re.sub("fi", "fl", text)
```

Listing 6.3: Replacing Ligature Characters

After removing the ligature characters, the normalised, tokenised text is ready for the stopwords to be removed.

6.2.4 Stopword Removal

A stopword can be defined as a word that is commonly used within a sentence. However, stopwords in this study are standard information security terms that need to be filtered out to enhance the quality of the topics. This was a delicate tweaking process. Only after the data were tokenised and viewed by the researcher a call was made to include several terms. The additional terms that was added was appended to the stopword list as seen in Listing 6.4, the total number of stopwords totaled 282. After the filler words were excluded from the corpus, the next step in the pipeline is to stem the words.

```
stop_words = stopwords.words('english')
stop_words.extend(['used','using','jam','found','plays','information','security','network', 'technology', 'bgp'])
```

Listing 6.4: Stopwords code

6.3 Stemming

As mentioned in Chapter 5, Section 5.2.2, the Porter stemmer is used in the research (Porter, 1980). The goal of a stemmer is to reduce the forms of a word to the common base word. Stemmers usually cut off the end part (affixes) to achieve above goal. Stemmers are easy to implement and do not need time or resources to complete the task. This research looked at high performance, low time, or resources spent on normalisation of words. Stemming was selected to be used in this study based on its simplicity and the time it takes. In Listing 6.5, we display the code snippet showing how the stemmer was used.

```
stemmer = PorterStemmer()
def stem_words(text):
    """
    Function to stem words, so plural and singular are treated the same
    """
    try:
        text = [stemmer.stem(word) for word in text]
        text = [word for word in text if len(word) > 1] #make sure we have no 1 letter words
    except IndexError: # catch the exception
```

```
        pass
    return text
```

Listing 6.5: Stemming the corpus

In Listing 6.5, the core code was to instantiate the Porter stemmer and feed the data through it. We also encountered some words like 'eod', which broke this and needed to insert a try catch, to make the code continue running.

Word	**Stemmed word**
informational	inform
translations	translat
evaluating	evalu
itemisation	item
awareness	awar
reference	refer
plotted	plot

Table 6.1: Stemming words from the data

Some of the words that were found in the data along with their root forms can be viewed in Table 6.1. The Porter stemmer takes the original word back to its root form.

In the next section, getting topics for the model will be discussed.

6.4 Topics for the model

This section will be a in-depth continuation of Chapter 5, Section 5.3.3. The bag of words was the step taken between normalising the data and feeding it into a topic modeling algorithm. Later in this section, the topic modeling technique and its parameter refinement will be discussed.

6.4.1 Bag of Words

For the topic modeling technique to compute the latent topics, it needs to understand the document. Before the text gets to the topic modeling technique, the text is just tokenised and normalised, thus creating the need to represent the text in a manner that the topic modeling technique can interpret it. The most common technique in natural language processing and information retrieval is the bag of words (BoW) model.

The bag of words (BoW) model can be used to extract features from the text. A BoW model is a way to simplify the representation of words in a document. Note that it is called a 'bag of words', the order of the words or sentence of the words does not bear merit. The BoW model is only interested in known words in the document, and the rest is discarded. Goldberg (2017) mentions that documents are similar if they have similar content. The BoW model can be simple to use or rather complex in determining the vocabulary of known words and how to score the reoccurring words.

Determining the vocabulary will be shown below. Listed are snippets from a paper dealing with Phishing attacks occurring through email and social networking sites. Each sentence will be handled as a document. Ignoring case and punctuation, the sentences are:

- Phishers continuously seek new methods...
- Conducting phishing solely through email...
- Social network phishing and discusses...

After removing stopwords. The unique words here are:

1. 'Phish'
2. 'Method'
3. 'Conduct'
4. 'Email'
5. 'Social'
6. 'Network'

This is a vocabulary of six words from a corpus of 15 words. Scoring the words can commence once the vocabulary has been selected. There are two scoring approaches one can follow. They are:

1. Counting the number of times the word is used in the corpus.

2. Frequencies can be calculated by how frequently each word is used out of all the other words.

This research employed the first scoring approach by counting the words. A library called Gensim with a function of doc2bow was used to count the number of times the word occurs, converts the word to a word id and then returns the result as a sparse vector (Rehurek & Sojka, 2010). Scoring the previous three documents against the vocabulary of six words will look as follows:

```
 Phishers continuously seek new methods = [1, 0, 0, 0, 0]
 Conducting phishing solely through email = [1, 1, 0, 0,
1]
 Social network phishing and discusses = [1, 1, 1, 0, 0]
```

Listing 6.6: Scoring the documents

Each sentence in Listing 6.6 was compared to the tokenised and normalised list. In conclusion, the BOW model disregards context, the meaning of the words, and lastly, the order in which the words appear. This gives the BOW model its strength because the more similar words are, the more similar documents are to each other. In the next subsection, we will discuss how the vector (BOW) is used in the topic modeling technique.

6.4.2 Topic modeling parameters

In Chapter 3, this study discussed what latent dirichlet allocation is and how it works. In this section, we will discuss how the LDA algorithm parameters work, on a deeper level. It should be said that in latent dirichlet allocation the order of the words do not matter, because of the bag of words model used in the study.

The dataset consists of several documents, and a document is a distribution over topics. Each topic within the document is a distribution over words that is then added to a vocabulary. Latent dirichlet allocation is a probabilistic model that identifies hidden variables within text. To infer such variables using LDA, the algorithm has parameters that steer the inference process. These parameters include the number of topics, chunksize, corpus, minimum probability, id2word, alpha, and beta. The last two parameters, alpha and beta, are known as hyperparameters. All of the parameters will now be defined:

1. Number of topics – the number of topics to be extracted from the training set.
2. Chunk size – the number of documents that are processed per training cycle.
3. Corpus – this is the stream of documents that have been transformed into a vector.
4. Minimum probability – inferred topics that have a probability score less than this threshold will be filtered out.
5. Passes - the number of passes the corpus goes through during training.
6. Id2Word – the main feature is mapping word id to words. It also help to determine the vocabulary size.
7. Alpha – a high value means that text will be represented by more topics and the inverse also holds true. Low value means that the text will be represented by fewer topics.
8. Beta – a high beta value means that the topics are represented by more words.

As seen in Listing 6.7, getting the best topics from the algorithm does require fine tuning. Most of the parameters have previous workings, which influence the quality of the topics inferred by die LDA algorithm, for example, with pre-processing it is garbag- in; garbage-out. Do the number of topics capture the total number of inferred topics? See in Listing 6.7 the parameters and values that this research used.

```
def train_lda(data):
    num_topics = 10
    chunksize = 300
    dictionary = corpora.Dictionary(data[’tokenized’])
    corpus = [dictionary.doc2bow(doc) for doc in data[’
   tokenized’]]
    t1 = time.time()
    ten = LdaModel(corpus=corpus, num_topics=num_topics,
   id2word=dictionary, chunksize=chunksize,
   minimum_probability=0.0,iterations=100)
    t2 = time.time()
    print("Time to train LDA model on ", len(df), "articles:
   ", (t2-t1)/60, "min")
    return dictionary,corpus,ten
```

Listing 6.7: LDA Parameters

When experimenting with the parameter, the researcher found that it was not that complex to obtain good topics with minimal tweaking of the parameters. However, one parameter, the number of topics, is one of the critical parameters that has the most influence over the quality of the model. Selecting the number of topics will now be discussed.

6.4.3 Selecting the Number of Topics

Finding the optimal number of topics for the LDA algorithm to provide better interpretability is a rather daunting task. There are two techniques to achieve the main goal of selecting the number of topics to be inferred. First, after running the LDA algorithm with K number of topics, one can use topic coherence, which scores a single topic by measuring the semantic similarity. The measurement is made in seeing how the topic is similar to the high-ranking words in the topic. Topic coherence helps to seek the difference between semantically inferred topics and topics that are created through statistical inference.

```
10topics = CoherenceModel(model=10LDA,texts=listt,dictionary=
    dictionary,coherence='c_v')
0.36285839731827135 = 10 topics
0.32723852556113137 = 15 topics
0.358826134976557   = 20 topics
0.33560120776491093 = 25 topics
0.33958962920470964 = 30 topics
0.3203274912859999  = 35 topics
```

Listing 6.8: Topic coherence

As seen in listing 6.8, 10 topics is a coherence model that uses the LDA model with the number of topics set to 10 and increments by 5 per output. Below that is the coherence score of a range between 10 to 35 topics respectively.

(Stevens, Kegelmeyer, Andrzejewski, & Buttler, 2012) argues that the higher the coherence score the more semantically similar the topics are. However, the first ready should normally be ignored and the next highest value should be used. Based on this, the researcher used 20 topics to infer from the LDA model.

Lastly, the other technique is more observation based called Eye-balled method. The technique consists of two main parts. One part is looking at the output of the LDA algorithm and seeing if the keywords in the topic are actually making sense. The other part pyLDAvis must be looked at. PyLDAvis is a visualization tool used to visualize the output of the LDA algorithm. In listing 6.9 topic 0 can be represented as: bank, mobile, similar, compute, website and much more. They are ranked from higher to a lower number between 0 and 1. Closer to 1 means that the weight is reflecting the importance of a keyword in that topic.

As seen in Listing 6.8, 10 topics is a coherence model that uses the LDA model with the number of topics set to 10 and increments by 5 per output. Below that is the coherence score of a range between 10 and 35 topics, respectively. Stevens et al. (2012) argue that the higher the coherence score, the more semantically similar the topics are. However, the first ready should normally be ignored and the next highest value should be used. Based on this, the researcher used 20 topics to infer from the LDA model.

Lastly, another technique, which is more observation-based, is called the eye-balled method. The technique consists of two main parts. One part is looking at the output of the LDA algorithm and seeing whether the keywords in the topic are actually making sense. The other part pyLDAvis must be looked at. PyLDAvis is a visualisation tool used to visualise the output of the LDA algorithm. In Listing 6.9, topic 0 can be represented as: bank, mobile, similar, compute, website, and much more. They are ranked from a higher to a lower number between 0 and 1. Closer to 1 means that the weight reflects the importance of a keyword in that topic.

```
[(0,
  '0.017*"bank" + 0.011*"mobil" + 0.007*"similar" + 0.007*"
   comput" + '
  '0.007*"websit" + 0.007*"student" + 0.007*"system" +
   0.006*"digit" + '
  '0.006*"vote" + 0.005*"technolog" + 0.005*"techniqu" +
   0.005*"framework" +
 '0.005*"measur" + 0.005*"domain" + 0.005*"phish" + 0.005*"
   toward" + '
 '0.005*"south" + 0.005*"develop" + 0.005*"distanc" +
   0.005*"engin"')]
```

Listing 6.9: LDA topic output

If the keywords in the topic contain no real words of value, the stopword list should be updated to remove such words. This is a common occurrence in domain-specific systems.

The last part of selecting the correct number of topics for the LDA algorithm is using visualisation tools such as pyLDAvis. This visualisation tool shows the soft clusters that the LDA algorithm provides. As seen in Figure 6.4, each bubble on the left hand side represents a topic. The larger the bubble, the more popular the topic. A good topic is considered to be displayed as a big bubble that does not overlap with the other bubbles. Therefore, a smaller sized bubble that overlaps with other bubbles, usually has too many topics.

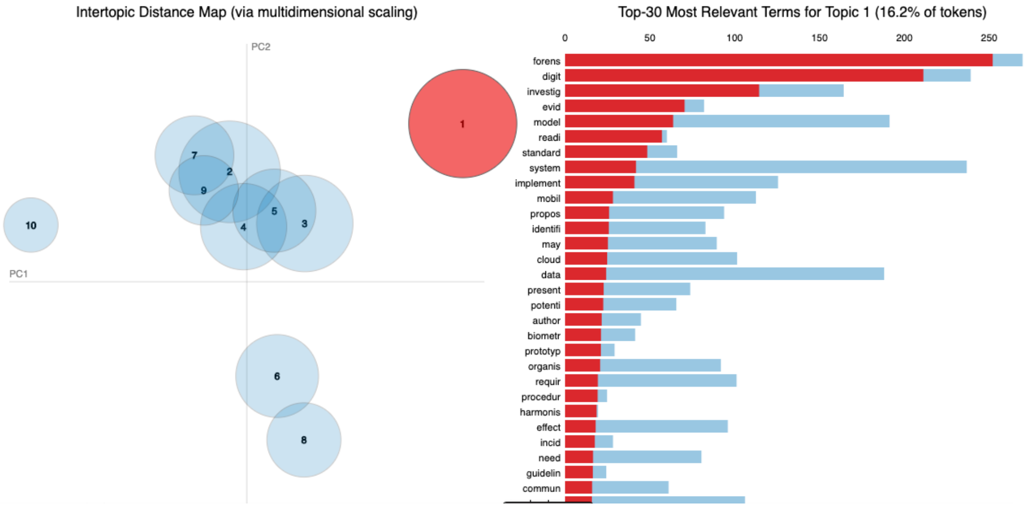


Figure 6.4: Visualisation package for LDA

Based on the coherence score and supported by the observations made in pyLDAvis, the researcher selected the LDA parameter number of topics as 20. In the next sub-section, the researcher will discuss how the similarity was calculated between the documents.

6.5 Similarity between documents

One of the main goals of this study is to present similar papers to the readers. In order to do this, the data needs to be tokenised, normalised and cleaned. After the pre-processing pipeline, the tokenised data should be transformed into a vector and fed into the latent dirichlet allocation topic modeling algorithm. After a few iterations and fine tuning of the parameters within the LDA algorithm, the output data can then be used to obtain deeper findings.

The goal of obtaining in-depth information rests on the Jensen-Shannon Divergence algorithm. As seen in Figure 6.10, we were computing the Jenson-Shannon Distance (JSD) using Scipy's entropy. JSD finds the distance between an input query (LDA topic distribution of a single document) and a big matrix (the whole corpus), as seen in line 2. In line 5 it returns an array of length m, where m is the number of document in the corpus.

Later in line 5, the square root of the Jensen-Shannon Divergence is returned because the square root of Jensen-Shannon Divergence is the Jensen-Shannon Distance. The smaller the JSD, the more similar the two topic distributions are to each other; in this case, the more similar two documents are to each other.

```
1 def jensen_shannon(query, matrix):
2     p = query[None,:].T
3     q = matrix.T
4     m = 0.5*(p + q)
5     return np.sqrt(0.5*(entropy(p,m) + entropy(q,m)))
```

Listing 6.10: Implementation of Jensen-Shannon similarity

The Jensen-Shannon function in Listing 6.10 is then implemented in Listing 6.11, where it actually computes the JSD and returns the top k smallest distances. These Jensen-Shannon distances were then used to find the recommendations of academic papers.

```
def get_most_similar_documents(query,matrix,k=10):
    sims = jensen_shannon(query,matrix)
    return sims.argsort()[:k]
```

Listing 6.11: Jensen-Shannon function

As mentioned in Chapter 3, the dataset was split into a training set and a testing set. The testing set comprised only one document. The document

Title of test paper	**Tokenised paper**
Testing the harmonised digital forensic investigation process model using an android mobile phone	['test', 'harmonis', 'digit', 'forens', 'investig', 'model', 'android', 'mobil', 'phone']

Table 6.2: Test paper and tokenized output

was kept aside while training the model. The test document was selected at random using a simple Python random function. The title of the test paper can be seen in Table 6.2, alongside the tokenised words from that one document corpus. Interpreting the tokenised data, a fairly good guess can be made of the content of the paper. The researcher translated the tokenised data as testing a model in the digital forensic field, looking at mobile phones, Android more specifically.

Looking at the JSD output of the two topic distributions, whole corpus and the one document, a recommendation can be made. In Listing 6.3 we can see that the scores are sorted in ascending order, ranging from 0,426 to 0,603. The smaller the score, the more similar that specific document is to the test document. However, no hard-coded threshold would have made a difference in identifying the top most similar documents. These most similar documents were, in fact, closely similar to the test document.

6.6 Summary

This chapter discussed the prototype development. It outlined the importance of having a good pre-processing pipeline in place. This entailed feeding the data that was extracted from PDFs into a JSON format file and stored on Github for easy access ability. After the data was tokenised and pre-processed, it was transformed into a vector so the LDA algorithm could use the data. Through the iterations of fine tuning the parameters of the LDA model, the number of topics were chosen (20 topics) and fed into the Jensen-Shannon Divergence similarity model and the papers were all ranked, based on similarity scores. Smaller similarity scores indicates that a particular paper is very similar to the test paper. The next chapter discusses the lessons learned while developing the prototype.

Title of papers	**Scores**
The design of a wireless forensic readiness model (WFRM)	0.426
Mobile forensics using the harmonised digital forensic Investigation process	0.452
Enhancing digital business ecosystem trust and reputation with centrality measures	0.498
Mobile cyber-bullying: A proposal for a pre-emptive approach to risk mitigation by employing digital forensic readiness	0.497
Remote fingerprinting and multisensor data fusion	0.508
Real-time distributed malicious traffic monitoring for Honeypots and network telescopes.	0.509
Harmonised digital forensic investigation process model	0.549
Digital forensic readiness in the cloud	0.579
Towards a digital forensic readiness framework for Public key infrastructure systems	0.592
Bimodal biometrics for financial infrastructure security	0.603

Table 6.3: Similarity scores of the most similar academic papers

Chapter 7

Lessons learned

The previous chapter covered the steps taken to build a prototype of the model presented in Chapter 5. The purpose of the prototype was to act as a driver to test various techniques. The prototype was refined based on similarity scores.

In this chapter, the algorithms and techniques are discussed in greater detail. The chapter will start off by discussing the rationale when picking pre-processing techniques, sparking the discussion around the gensim LDA algorithm, whether the antecedent algorithms and techniques influenced the similarity measures and lastly, evaluating three topics.

7.1 Pre-processing

In this section, the aim is to discuss pre-processing techniques and the significance of choosing the correct combination. Every natural language processing project has its own set of pre-processing techniques; this study is no different. The combination of techniques were derived from literature, as discussed in Chapter 3.

At first, combining the set of techniques was simple to do. However, by the time the data was at the end of the pipeline, quality inconsistencies could be seen in Listing 7.1. This established the challenge to clean the data to a satisfactory level.

```
'unfortunately','huge','used','myburg@gmail.com','information
    ',
'security','aodv','A12fD','derive'
```

Listing 7.1: Custom list of stopwords

The removal of stopwords can be bundled into a three-part process: (1) regular expressions, (2) removing the stopwords, and lastly, (3) removing the ligature characters. Throughout the process, three lessons were learned, not only during the experimentation stage of the study, but also during the refinement phase.

During the experimentation stage of the study, the first takeaway was that being too strict with the words in the stopword list would have a negative impact on the quality of the topics. However, leaving out stopwords on purpose would render the same results.

Lesson 1 (Goldilock's dilemma)**.** *The identification and removal of stopwords is a very important part of the pre-processing pipeline. Removing too many domain-specific words negatively influences the quality of the topics, and ultimately influences the similarity scores.*

Removing too many stopwords would increase the risk of losing the context of the topics, since the approach of this study is to employ unsupervised learning methods, which makes context key.

Leaving too many stopwords in the pipeline, will make the trained model convoluted. The model will then soft cluster topics, which brings no significant value and will result in inaccurate recommendations.

The optimal number of domain-specific words which are in included in the stopword list should solely be based on experimentation.

Building on the customisation of the stopword list, a second lesson was learned. Researchers used different tools for typesetting, where ligatures occur. Modern typesetting tools use updated fonts, which do not use ligatures. The problem with it was that the pre-processing techniques could not identify the ligatures and let them passed unscathed, ending up in the dataset.

Lesson 2 (Problematic ligature characters)**.** *Researchers using different tools for typesetting can result in the pre-processing pipeline excluding words containing ligature characters.*

Reducing dimensionality by using pre-processing did play a major part in the time performance of the algorithms. It decreased the time it took to train the models. There were 5446 unique words in the dataset without removing stopwords. Furthermore, removing more domain-specific words lowered the unique words count to 3376. Even though the unique words were reduced, we still needed to apply LDA to the words.

7.2 LDA Parameters

As described in Chapter 6, Section 6.4.2, latent dirichlet allocation has five parameters and two hyperparameters, which need refinement in order to achieve quality topics. However, only four parameters and two hyperparameters will be considered as they are the only parameters which need refinement.

The quality of the topics were measured in two ways: by using topic modeling evaluation techniques and human intervention by the researcher. First, the coherence score and perplexity were calculated. It should be noted that a higher coherence score indicates quality topics and simultaneously, a low perplexity score also indicates quality topics.

Manual intervention has been used in this study to validate the recommendations made. This was done by looking at the test paper, identifying a few dimensions like:

1. What is the main topic discussed in this paper?
2. What are the secondary topics discussed in this paper?
3. What is the argument made by the researcher in this paper?

Once the dimensions were identified, a similar exercise was done on the training set. The recommendations was an output of 5 most similar papers, ranked from the lowest Jensen-Shannon Divergence score (most similar), to the highest JSD score.

The researcher then looked at the 5 recommended papers and compared the above-mentioned dimensions to the test paper. In most cases some of the recommendations were similar. Once the researcher identified some recommendations to be not similar, the researcher looked at how changing the LDA parameters would affect the coherence, perplexity scores and, ultimately influence the recommendations.

Human intervention was needed to validate if good coherence and perplexity scores translated into good recommendations. The observation was that for some it was true, however, some recommendations with good coherence and perplexity scores were not good recommendations. The researcher looked at the coherence and perplexity scores for the LDA parameters of the recommendations.

As mentioned, the number of papers that are recommended is 5. This number could be increased to see when the recommended papers stopped being similar however, the study focused on getting the 5 most similar papers and see how similar they are to the test paper.

Lesson 3 (Manual intervention)**.** *When evaluating the quality of the topics generated by the model, even though evaluation techniques like coherence scores or perplexity indicate good topics, manual intervention is still needed to validate them.*

Every LDA model first needs to establish a base model to gauge a baseline. As mentioned in Chapter 6, Section 6.4.2, the number of topics was chosen as 20. Since establishing a base model needs human intervention to identify and to separate good results from the rest, most of the initial parameter values were chosen based on previous literature and through experimentation (Baghel & Dhir, 2010). Optimising for these parameters may not yield humanly interpretable results.

The LDA parameters needed to be constant throughout the refinement process. The only parameter that changed was the one that was closely monitored. The parameter values were:

1. Number of topics - 20
2. Chunksize - 20
3. Passes - 20
4. Minimum probability - 0.001
5. Alpha - 0.1
6. Beta - 0.9

In the rest of the section, the lessons will be discussed that were learned regarding parameter refinements.

Number of passes	2	20	200
Time to train the model	0.020 Min	0.138 Min	1.185 Min
Perplexity	-7.574	-7.400	-7.3099
Coherence Score c_v	0.3056	0.4411	0.4668

Table 7.1: Results on the number of passes

7.2.1 Passes

Exploring the effect that passes have on the model showed a number of interesting observations. When the number was set to 2 passes, the model trained within two seconds. The perplexity score was -7.574; this was by far the highest score from all of the tests. In addition, the coherence score was also low with 0.3056, which is significantly lower than the rest.

Next, the passes parameter was set to 200. This yielded better results than the previous one. However, the trade-off was that the model took 1.185 minutes to train. The perplexity score did improve by 0.3 to -7.3099, and the coherence score also improved to 0.4668.

Lastly, the researcher chose the value 20. The model trained significantly faster than the previous test. It took merely 8.28 seconds to train the model, which is a 758% increase on the previous test. Perplexity and coherence scores were -7.4 and 0.4411, respectively.

The lesson learned was that the performance gain with regard to model training speed was much better than the increase in quality of the topics. Passes parameter 2 and 200 both had trade-offs: quality and speed, as referenced in Table 7.1.

Lesson 4 (Quality over quantity)**.** *Increasing the number of passes does not automatically increase the quality of the topics.*

Through human intervention, the researcher evaluated the topics to see whether they were making sense. With the lower passes parameter set, it was observed that the topics could be read and identified. However, some topics were spotted containing words which does not fit the topic it was in.

7.2.2 Chunksize

As per the definition, chunksize refers to the number of documents to be used in each chunk. It is also one of the optional parameters in the gensim LDA

Chunksize	20	125	253
Time to train the model	0.083 Min	0.059 Min	0.135 Min
Perplexity	-7.377	-7.523	-7.422
Coherence score c_v	0.591	0.394	0.4674

Table 7.2: Results of adjusting the chunksize.

library. In principle, chunksize should not have an effect on the quality of the LDA topics. However, after testing the model with various sets of chunksize parameters, the numbers shows a different story.

As seen in Table 7.2, a chunksize of 20, 125, and 253 greatly affected the time taken to train the model, the perplexity value and lastly, the coherence score.

It was anticipated that reducing the chunksize, the model would take longer to train. Looking at the time values in Table 7.2, with a size of 20 the model trained in 0.083 minutes (4.98 seconds). Furthermore, changing the chunksize parameter to roughly half of the total number of documents, 125, the model trained in 0.059 minutes (3.54 seconds). Using 125 chunksize decreases the time the model took to train by 28.91%. In contrast to using 253 as the Chunksize, which includes all the documents in the testing set, the time it took to train the model increased to 0.132 min (7.92 seconds). Comparing the difference in time the model took between 20 and 254 chunksizes was 37.12%.

Moving to the perplexity scores of each iteration: 20, 125, and 253, respectively, a chunksize of 20 produced a perplexity score of -7.377, which was the lowest compared to all the others. This signifies that the model using a chunksize of 20 produced better topics then the rest. However, this means that it is only slightly better than using 125 and 253 chunksizes. A perplexity score of -7.377 is an increase of 1.9% and 0.6% compared to using the other chunksize numbers.

The coherence score of these tests proved that using a smaller chunksize does take longer to train the model; however, it does yield better results. As seen in Table 7.2, using a chunksize of 20 outperformed the rest by a big margin. Comparing the test sets, 20 to 125, and 20 to 253, there was a decrease in coherence score by 33.33% for the first comparison and a further decrease by 20.9% for the second comparison.

Probability	**Topic**
0.002	softlift
0.001	distanc
0.001	popul
0.001	residenti
0.001	dcss
0.001	profil
0.000	wherea
0.000	incom
0.000	strongli

Table 7.3: Using minimum probability as 0.00

The lessons learned when comparing these above-mentioned results lead to the following point. After each test set the researcher looked at the topics to evaluate the readability and to see whether the topics can be interpreted to something of value. However, after close inspection, no noticeable difference could be flagged.

Lesson 5 (Chunksize has little influence)**.** *Increasing the chunksize does not have a significant influence on the quality of the topics.*

7.2.3 Minimum-probability

As unwanted words filter through the pre-processing pipeline into the trained model, one of the ways to combat this is to have a minimum probability value set. Table 7.3 shows how using no minimum probability threshold can run the risk of picking up unwanted words, which translate to topics.

Table 7.4 details the results of testing different minimum probability values. Setting the various probability values did not yield any impact on the time it took to train all the models. One would expect to see the model being trained faster as the minimum probability value increased. This was, in fact, the case. Comparing the perplexity score of the three models revealed that increasing the probability value did not impact the perplexity score.

It was highly anticipated that minimum probability would have a direct impact on the coherence score since it measures the quality of the topics. The take-away from these refinements was that, as the time and perplexity score were not affected by the parameter, the coherence score was.

Minimum Probability	0.00	0.001	0.1
Time to train the model	0.081 Min	0.078 Min	0.077 Min
Perplexity	-7.381	-7.385	-7.381
Coherence score c_v	0.444	0.536	0.6203

Table 7.4: Results of testing minimum probability values.

The value of 0.1 produced the best coherence score; however, human intervention needed to take place. Increasing the value to 0.1 meant that infrequently used terms would be disregarded. In the context of this dissertation, the data set consists of abstracts which do not contain a large text.

In light of the results shared in Table 7.4, the researcher chose the minimum probability of 0.01 to be used in the main model. The lesson learned was that if the parameter minimum probability was tweaked too high, most of the words that would add context to each set of topics would be omitted. This being an unsupervised machine learning proposed model, running certain risks would be unavoidable. It was all about getting the threshold.

Lesson 6 (Probability Dilemma)**.** *If the minimum probability is set too high, the topic model runs the risk of not including certain topics. Smaller documents often only have a few sentences, which influences the probability of word occurring. Therefore, the minimum probability cannot be too high.*

7.2.4 The number of LDA topics

In Section 6.4.3, the systematic process was discussed that was used in determining the number of LDA topics. In addition to that, lessons were learned throughout the process and will be discussed in this section.

The tests varied from 10 topics to 35 topics. In Listing 6.8, the fluctuating topic coherence scores were a quantitative indication of the most suitable number of topics. However, the process still needed human intervention to confirm that the topics were making logical sense.

Keeping in mind that the corpus (ISSA Conference papers) was rich in topics, the conference papers contained about 15 primary and secondary topics; essentially, picking 10 or 15 LDA topics to try to represent the topics from the text failed.

Number of LDA topics	5	20	50
Time to train the model	0.053 Min	0.079 Min	0.125 Min
Perplexity	-7.341	-7.488	-7.611
Coherence score c_v	0.34	0.35	0.41

Table 7.5: Results of testing different number of LDA topics.

The first lesson that was learned in this process was that a low number of LDA topics will not represent the true topic depth of the text. In Listing 7.2, only five topics were chosen and we can derive the general themes they represent. Running the model again will render different results; thus showing that lowering the number of topics will result in wasting potential LDA topics and ultimately skew our recommendation at the end of the pipeline.

It is understood that every time the model runs, there is a chance that it will provide slightly different results. However, increasing the number of LDA topics would provide a safety net that the strong topics will always prevail.

```
Topic 1 - 0.016 system + 0.016 user + 0.011 social + 0.011 privaci + 0.010 manag
Topic 2 - 0.027 south + 0.024 risk + 0.016 africa + 0.015 popi + 0.013 govern
Topic 3 - 0.063 forens + 0.054 digit + 0.042 investig + 0.020 data + 0.014 databas
Topic 4 - 0.030 network + 0.025 mobil + 0.022 attack + 0.019 detect + 0.018 applic
Topic 5 - 0.027 cloud + 0.022 servic + 0.017 comput + 0.015 pattern + 0.012 face
```

Listing 7.2: 5 Number of topics

In Table 7.5, three tests were undertaken, which included time taken to train the model based on the number of topics, the perplexity values per number of topics and lastly, the coherence score of each.

It was to be expected that the time would increase as the number of topics grew in size. The time increased by 49.05%, and then by 58.22% starting from five topics to 20 and ending off with 50 topics.

Furthermore, the lesson learned from observing the perplexity score over time in Table 7.5, was that with every step from five to 20 to 50 topics, the score gradually decreases. It is to be expected that the perplexity score will flatten out as the number of topics increase. However, as stated in a previous lesson learnt, human intervention will be needed to validate whether the topics still make semantic and logical sense.

Lesson 7 (Flatten the curve)**.** *The perplexity score will flatten out as the number of topics gradually increase.*

Looking at the coherence score, which also gradually changes over time, human intervention is still needed. The coherence score increases with 2.9% and 17.14% with each increase of topics.

7.3 Latent dirichlet allocation hyperparameters

In this section the lessons learned regarding the two hyperparameters, alpha and beta, will be discussed. The hyperparameter beta will be also referred to as eta (gensim).

Figure 7.1 illustrates the interconnectivity between documents, topics and words, furthermore, showing that favouring one of these two hyperparameters will result in either of two things:

- High alpha - the documents will have high number of topics contributing to them.
- High beta - the topics have a high number of words contributing to them.

In the rest of this section, the lessons learned for both of these hyperparameters will be discussed.

7.3.1 Dirichlet hyperparameter alpha

Choosing a value of alpha or beta (eta) can be a tricky task, since these two hyperparameters have an direct impact on the topic modeling results. A high alpha means the documents contain more topics.

The researcher experimented with various alpha values. These values were 0.01, 0.1, and 0.5. It was discovered that using the above-mentioned

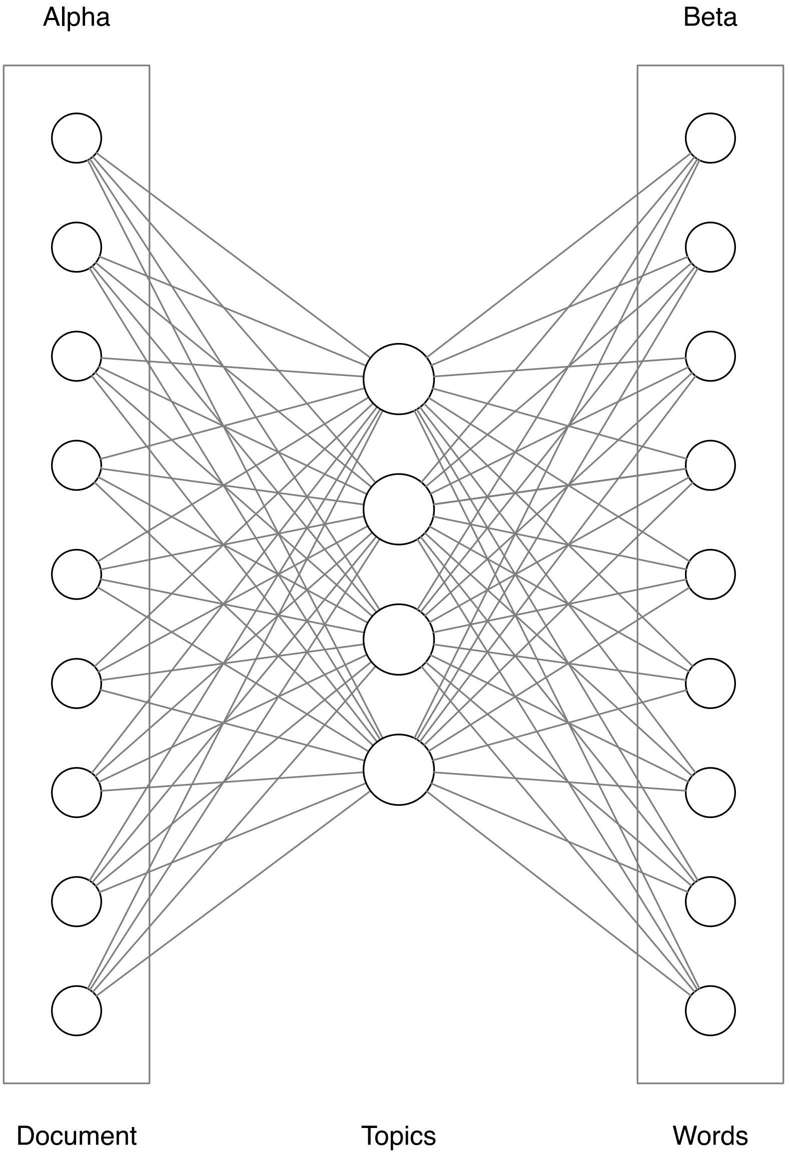


Figure 7.1: Alpha and beta representation

values, the actual prototype was displaying run time divided by zero errors. The cause of the errors was linked to the alpha and beta values. The model was producing explicit zero results.

As Wallach, Mimno, and McCallum (2009) mention, an alpha asymmetric is good, but an asymmetric beta is not. It was because of the run time errors that it was decided to set the parameters to auto. This means that the model learns an asymmetric statistical inference, or prior for short. The hyperparameter beta will be discussed in the next sub-section.

7.3.2 Dirichlet hyperparameter beta

It was also found that experimenting with various beta (eta) values rendered the same run time errors observed with the alpha testing. The researcher also decided to set the hyperparameter to auto, which learns asymmetric prior from the data.

This approach of setting the hyperparameters to auto, as stated, was found to benefit the topics generated by the LDA model. Giving the ability to the data to learn their own prior meant that it should deliver similar results each time the model ran. Capping or limiting the hyperparameters would result in run time errors and skewed results.

As mentioned by Griffiths and Steyvers (2004), when working with scientific documents, it is best to use a larger beta (eta) value. The reason is that it could lead to the model containing a smaller number of topics and could cover more scientific concepts. This is in contrast to using a smaller beta (eta) value, which would produce more topics that would address specific concepts.

In the next section, similarities will be discussed between the probability distributions.

7.4 Similarity between probability distributions

In this section, lessons learned will be shared on the process of finding the similarities between the documents.

In Section 3.2.3, Jensen-Shannon Divergence (JSD) was mentioned to be the improved method used to measure similarity between two probability distributions. The researcher used the Jensen-Shannon Divergence method to calculate the distance between the two probability distributions. In recent years, research has indicated that JSD still has relevance today (Tong & Zhang, 2016; Giles, Ang, Mihaylova, & Arvaneh, 2019; He, Chen, Du, & Jiang, 2015; Chu & Li, 2010).

As there are no parameters with Jensen-Shannon Divergence, this made it easy to implement it into the prototype and ultimately, to look at the end results. Since the goal of this research was not to compare other similarity measurement methods, the researcher only considered JSD.

In the next section, evaluating the output of the similarity of the probability distributions will be discussed.

7.5 Evaluation

In this section the observations will be discussed between the test document and training documents. Observations will be shared to understand whether and what made the similarity list good or bad. In the next subsection a single test document with its corresponding list of similar documents will be discussed.

The layout of the subsection will include observations on what made a topic like digital forensics and similar documents good, what LDA parameters could contribute to the change of similar documents, and how many documents matched. This structure will continue with what was considered a good match and what a bad match.

In addition to the layout of the subsections, two topics were selected to demonstrate the prototype and model. More topics were used in the experiment, however, these two topics were selected to illustrate the lessons which were learned during the study. Data related to the the other topics can be viewed in Appendix A.

7.5.1 Digital Forensics

The title of the test document was ‘Towards a Framework for Enhancing Potential Digital Evidence Presentation’. The contents of the document covered

JSD SCORE	TITLE
0.5372	Towards a digital forensic science
0.6170	A model for secure value-added service subscriptions in cellular networks
0.6172	POPI act - Opt-in and opt-out compliance from a data Value chain perspective: A South African insurance industry experiment
0.6179	Team formation in digital forensics
0.6306	The current state of digital forensic practitioners in South Africa

Table 7.6: Digital forensic similarity

working towards a framework to develop methodologies and specifications. The ultimate goal is to enhance the presentation and interpretation of any legal proceedings effectively.

The recommendation list of documents of the above topic consists of five documents. The list of recommendations are listed in Table 7.6. Observing the list, one could argue that the topic, digital forensics, was a good topic to test with.

The researcher observed that one of the reasons that digital forensics was a good topic to test with was that there were multiple documents covering the same topic. This meant that the LDA model had more training data. Having multiple documents covering the same topic provided more words for the bag of words representation of the text.

Another observation was that since Jensen-Shannon Divergence had no real parameters to adjust, little difference could be made in the greater scheme of things. The majority of the tweaking and working must be done before getting to the Jensen-Shannon Divergence. The tweaking was done in the text pre-processing, text representation, and LDA model phases of the experiment.

Lesson 8 (JSD no parameters)**.** *Jensen-Shannon Divergence (JSD) does not have parameters to refine, thus putting more emphasis on the refinement of the pre-processing and topic modeling components.*

In terms of changing the LDA parameters, it was observed that increasing the chunksize of the LDA model to cover all the documents in the training set yielded good results for this specific topic. In addition, increasing the number of passes during training also yielded good results.

In the next subsection, the topic privacy, similarity list and observations will be discussed.

JSD SCORE	TITLE
0.5612	A profile of the distance computing student softlifter
0.5820	CDMA in signal encyption and information security
0.6670	Context aware mobile application for mobile devices
0.6671	Towards a framwork for a network warfare capability
0.6676	Adaptable exploit detection through scalable netflow analysis

Table 7.7: Privacy similarity

7.5.2 Privacy

The test document that was used had the title ‘Computer monitoring in the 21st century workplace’ which laid the foundation for a workplace privacy policy that protects employees.

The recommendation list has also provided the top five documents which it considers very similar to the document above. As viewed in Table 7.7, the titles of the documents do not indicate any privacy topics.

The main observation for this subsection is that words that would be more easily identifiable in the LDA model became lost in the pre-processing and LDA pipeline.

The other observation was that once the number of topics increased, it created room for the other topics also to be included. Those topics were previously the second- or third-tier topics in the LDA model.

Lesson 9 (More is not better)**.** *Increasing the number of topics increases the risk that topics will be included that do not provide any significant value.*

As simple as it may sound, compared to digital forensics, there were not many documents which dealt with privacy and workforce topics.

7.6 Summary

In this chapter, we discussed several themes while developing the prototype. What changes were needed to be made for the prototype to perform at its maximum? The other theme was the lessons that were learned while administering the changes.

The lessons learned from each component, as indicated in Figure 6.1, are summarised and shared in Table 7.8. The conclusion that is drawn is that pre-processing is certainly one of the most important legs in the pipeline and needs to have most of the time dedicated to it. Each component of the pre-processing process needs to be closely considered. Removing too many words can cause documents not to any similarity to others and leaving too many words can make the LDA model too sensitive.

The experimentation, analysis, and discussion enabled the creation of the model used in Chapter 6. The model outlines the steps that were needed to achieve a recommendation list. It also opened discussions so that researchers can further the research by using additional techniques and algorithms, which will be discussed in the next chapter.

Lesson name	**Lesson**
1. Goldilock's dilemma	The identification and removal of stopwords is a very important part of the pre-processing pipeline. Removing too many domain-specific words negatively influences the quality of the topics and ultimately, influences the similarity scores.
2. Problematic ligature characters	Researchers using different tools for typesetting can make the pre-processing pipeline exclude words containing ligature characters.
3. Manual intervention	When evaluating the quality of the topics generated by the model, even though evaluation techniques like coherence scores or perplexity indicate good topics, manual intervention is still needed to validate them.
4. Quality over quantity	Increasing the number of passes does not automatically increase the quality of the topics.
5. Chunksize has little influence	Increasing the chunksize does not have a significant influence on the quality of the topics.
6. Probability Dilemma	If the minimum probability is set too high, the topic model runs the risk of not including certain topics. Smaller documents often only have a few sentences, which influences the probability of a word occurring. Therefore, the minimum probability cannot be too high.
7. Flatten the curve	The perplexity score will flatten out as the number of topics gradually increase.
8. JSD no parameters	Jensen-Shannon Divergence (JSD) does not have parameters to refine, putting more emphasis on the refinement of the pre-processing and topic modeling components.
9. More is not better	Increasing the number of topics increases the risk that topics will be included that do not provide any significant value.

Table 7.8: Summary of the lessons learned

Chapter 8

Conclusion

Based on the discussion in the previous chapter, it is clear that using the various natural language processing and text mining techniques outlined in Chapter 6 would indeed be possible. It was apparent that adjusting the parameters of the prototype also played a major role in the quality of the topics. Later, such topics directly influenced the recommendation list.

This chapter provides a summary of the study to conclude the findings of the study. The next section will discuss how the research objectives were reached. After that section, suggestions will be made for future research.

8.1 Revisiting the research objectives

This section facilitates a discussion on how the research objectives were met. As mentioned in Chapter 1, the primary research objective was to ***Develop a model to recommend related research papers.***

Achieving the primary objective depended on meeting the secondary objectives listed in Chapter 1.

SO 1: *To identify recommender systems techniques and how they are used.*

This secondary objective was met by surveying the literature about recommender systems. As the goal of this secondary objective was to map and understand which techniques are used in recommender systems and how researchers utilise it, I looked at the various approaches that not only fit the use case of the study but also at which are relevant.

It was decided that the content-based filtering approach would the best to use since no assumptions are made about user activity. In addition to this, CBF does not care about user ratings; it looks at the content of the documents. It was critical to look for approaches that do not rely on user activity and user ratings. Chapter 2 was dedicated to introducing the concepts of how modern information systems (Recommender Systems) work and how they have evolved from the past to recent years. A continuation of the introduction to the fields was found in Chapter 3.

SO 2: *To identify machine learning techniques that assist with the recommender task.*

This secondary objective was met by surveying several machine learning techniques. The goal of this objective was to identify and understand better what machine learning techniques there are, and how they tie into recommender type systems.

Machine learning in Section 3.1, natural language processing in Section 3.3, and topic modeling in Section 3.4 paved the way to understand the technology that would be used in this study.

The selection of the topic model algorithm that would be most suitable came from literature. More specifically, in Section 3.4 it was discussed that latent dirichlet allocation (LDA) was a very popular choice for building such topic models. Furthermore, Section 3.4.1 explains how LDA works, by computing hidden topic structures from documents.

The main research objective was met by consolidating **SO 1** and **SO 2** to form a conceptual model. The model was refined by developing a prototype.

To develop such a model, some guidance was needed. The researcher used the data analytics lifecycle process to harness work done by secondary objectives one and two. Further refinement was needed on the model to display research rigour.

The development of the model was documented in Chapter 5, accompanied by the development of the prototype in Chapter 6. A prototype was developed to refine, ensure the amendments to be done, and showed whether the model is applicable and feasible. The model demonstrated applicability since it required domain-specific data for training and this was done using Information Security South Africa conference data. In addition, the model also demonstrated feasibility with its ease of using recommender system and machine learning techniques.

In the next section, a reflection on the model will be discussed.

8.2 Reflection on the Model

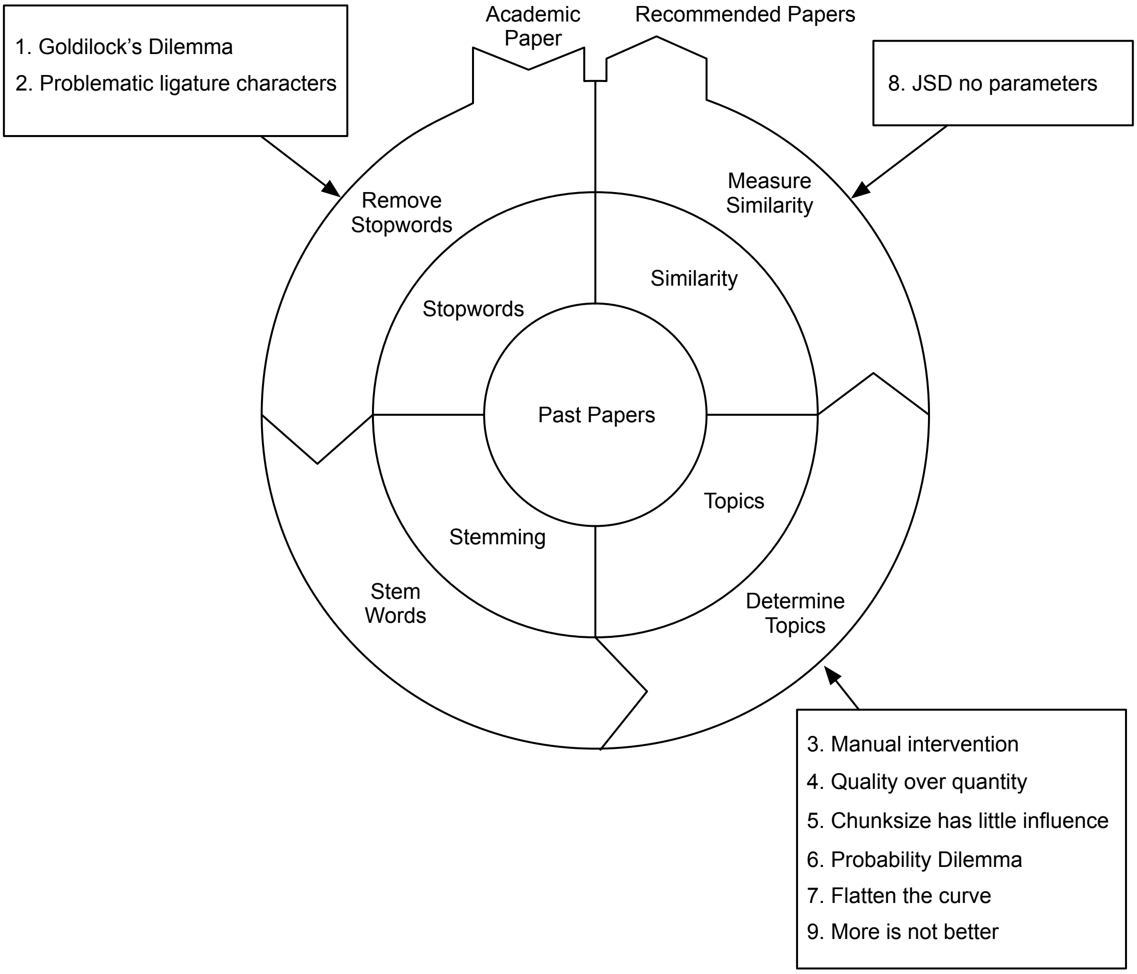


Figure 8.1: Mapping lessons learned to the model

This section reflects on the model and the accompanying prototype. The model was developed to identify related research papers. The model specified the process that needs to be followed and the prototype showed that it is indeed feasible to implement. The prototype showed that it worked within the specific domain of information security papers, since we used Information Security South Africa conference papers from 2006–2016 as data. Addressing the main objective using the model and prototype, it was shown that it can be done, theoretically and practically.

The prototype served a dual purpose. The first was to learn technologies and to gain first-hand experience related to recommender systems. The second was to demonstrate its applicability by using various technologies and feasibility by using a information security specific domain.

The majority of the lessons learned exists in the topics quadrant of the model, as depicted in Figure 8.1. It was found that removing the stopwords and stemming the data meant that it was streamlined and the body of knowledge was well defined. Removing stopwords and stemming present two lessons to draw the attention of future researchers to the subtleties in the area. For testing similarity measures, only Jensen-Shannon Divergence was tested. Jensen-Shannon Divergence does not use parameters, and therefore had an big influence on the preceding steps.

This opens up opportunities for future research to be undertaken using different similarity measure technologies, which do contain parameters.

To get back to the majority of the lessons about determining the topics. Lessons four, seven and nine show us that if we can define the domain better, the model will perform better. A trade-off thus exists between accuracy and generality. In the prototype, the domain was limited to information security papers by using research papers from the Information Security South Africa conference.

Moving on to lesson three, manual intervention. It was found that evaluation techniques for machine learning algorithms are well defined and streamlined. However, human validation is needed, since bad topics still make it into the system, thus challenging future researchers to look for methods and techniques that eliminate such human intervention altogether.

8.3 Research Limitations

This study has some limitations that must be recognised. The first limitation is that of the sample size. At the time of data collection, the Information Security South Africa conference only had 254 research papers to use in the dataset. Of those 254 research papers, certain topics are only discussed in a limited number. If the topics are scarce during training, the recommendation task will be very difficult to complete. The goal was to train and test the prototype using one specific library of one conference.

The second limitation is that only one topic modeling technique (latent dirichlet allocation) was used. There were several reasons for this, one of which is that recent research suggests that when using content-based approaches, latent dirichlet allocation would be preferred. Considering time constraints, since this is a Master's study, we felt it right to go ahead with the one topic modeling technique.

The third limitation is that this study only uses Jensen-Shannon Divergence to calculate the similarity between the two data spaces (test set and training set). Similar to the second limitation, time did not allow us to venture deeper into different similarity measures. The fourth limitation includes not having a human factor to evaluate each phase individually. The phases which are referred to are the outputs of the topic modeling technique and also the end result of the similarity measurement.

Lastly, the fifth limitation is that this study did not compare how different recommender systems performed (optimisation and recommendations). This research study focused on documenting the process of using one recommender system and discussing the observations.

8.4 Suggestions for further research

This section is dedicated to outlining further research to be done on the limitations which were raised in the previous section. Future research could possibly be conducted using a conference that has more research papers at their disposal. It will provide more data to be clustered and will ultimately, better the similarity measures. This relates to the first limitation.

Reflecting on the second limitation, the study could include multiple topics modelling algorithms and compare the outputs. Examples of such algorithms are: non negative matrix factorisation (NMF) (Purpura, 2018), latent semantic analysis (LSA) (Qomariyah, Iriawan, & Fithriasari, 2019), parallel latent dirichlet allocation (PLDA) (Mukherjee & Poovammal, 2019) and Pachinko allocation model (PAM) (Koltcov, Ignatenko, Terpilovskii, & Rosso, 2021).

Similarly, the third limitation, using different similarity measures could lead to different recommendations. Furthermore, mixing different topic models with similarity measures could yield better results.

The fourth limitation, which includes human intervention to evaluate each phase, could speed up the model implementation, since one would be getting feedback every step of the way. Suggestions for further research can include looking for methods and techniques that eliminate manual intervention.

Reflecting on the fifth limitation, this study did not compare the performance and recommendations of different recommender systems. This limitation can be addressed in future research by considering alternative recommender systems. Examples of such recommender systems are: a hybrid model (Sharma & Singh, 2016) and model-based collaborative filtering (Naak et al., 2009). Future research could possibly be conducted using data sets which includes user-item pairing data.

8.5 Epilogue

This study identified that it is time consuming for researchers to look for related research papers. This problem was addressed by developing a model to recommend related research papers. Throughout the development of the model and prototype, it was evident that machine learning bridges the gap of spotting latent themes. However, Chapter 7, the lessons learned, shows that when other researchers endeavour to explore similar topics, a great learning curve awaits. The researcher hopes that this study will motivate other researchers to advance the research angle of the traditional topic of natural language processing and machine learning.

References

Abualigah, L. M., Khader, A. T., Al-Betar, M. A., & Alomari, O. A. (2017). Text feature selection with a robust weight scheme and dynamic dimension reduction to text document clustering. *Expert Systems with Applications*, *84*, 24–36.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*(6), 734–749.

Alhawarat, M., & Hegazi, M. (2018). Revisiting k-means and topic modeling, a comparison study to cluster arabic documents. *IEEE Access*, *6*, 42740–42749.

Al-Kumaim, N. H., Hassan, S. H., Shabbir, M. S., Almazroi, A. A., & Al-Rejal, H. M. A. (2021). Exploring the inescapable suffering among postgraduate researchers: Information overload perceptions and implications for future research. *International Journal of Information and Communication Technology Education (IJICTE)*, *17*(1), 19–41.

Alpaydin, E. (2010). *Introduction to machine learning* (2nd ed.). The MIT Press.

AlSumait, L., Barbará, D., Gentle, J., & Domeniconi, C. (2009). Topic Significance Ranking of LDA Generative Models. In W. Buntine, M. Grobelnik, D. Mladenić, & J. Shawe-Taylor (Eds.), *Machine learning and knowledge discovery in databases* (pp. 67–82). Berlin, Heidelberg: Springer Berlin Heidelberg.

André, Q., Carmon, Z., Wertenbroch, K., Crum, A., Frank, D., Goldstein, W., Huber, J., Boven, L. van, Weber, B., & Yang, H. (2018). Consumer

Choice and Autonomy in the Age of Artificial Intelligence and Big Data. *Customer Needs and Solutions*, *5*(1), 28–37.

Andrews, N. O., & Fox, E. A. (2007). *Recent developments in document clustering* (Tech. Rep.). Department of Computer Science, Virginia Polytechnic Institute & State University.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval* (Vol. 463). ACM Press New York.

Baghel, R., & Dhir, R. (2010). A frequent concepts based document clustering algorithm. *International Journal of Computer Applications*, *4*(5), 6–12.

Bagul, D. V., & Barve, S. (2021). A novel content-based recommendation approach based on LDA topic modeling for literature recommendation. In *Proceedings of the 6th International Conference on Inventive Computation Technologies, ICICT 2021* (pp. 954–961).

Bates, M. (1995). Models of natural language understanding. *Proceedings of the National Academy of Sciences*, *92*(22), 9977–9982.

Bawden, D., & Robinson, L. (2020). *Information Overload: An Overview.* Oxford University Press.

Belkin, N. J., & Croft, W. B. (1992). Information Filtering and Information Retrieval: Two Sides of the Same Coin? In *Communications of the ACM* (Vol. 35, p. 29-38).

Bengoetxea, K., & Gojenola, K. (2010). Application of Different Techniques to Dependency Parsing of Basque. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-rich Languages* (pp. 31–39).

Benson, E., Haghighi, A., & Barzilay, R. (2011). Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Vol. 1, pp. 389–398).

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning research*, *3*, 993–1022.

Bokde, D., Girase, S., & Mukhopadhyay, D. (2015). An Item-based Collaborative Filtering using Dimensionality Reduction Techniques on Mahout Framework.

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, *12*(4), 331–370.

Burkov, A. (2019). *The Hundred-page Machine Learning Book* (1 ed.). Kindle Direct Publishing.

Cantador, I., Bellogín, A., & Vallet, D. (2010). Content-based Recommendation in Social Tagging Systems. In *Proceedings of the fourth ACM conference on Recommender Systems* (pp. 237–240).

Celma, Ò., & Herrera, P. (2008). A new approach to evaluating novel recommendations. In *Proceedings of the 2008 ACM conference on Recommender systems* (pp. 179–186).

Chaney, A., & Blei, D. (2021). Visualizing Topic Models. *Proceedings of the International AAAI Conference on Web and Social Media*, *6*(1), 419–422.

Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L., & Blei, D. M. (2009). Reading Tea Leaves: How Humans Interpret Topic Models. In *Neural Information Processing Systems* (pp. 288–296).

Chen, C.-L., Tseng, F. S., & Liang, T. (2010). An integration of WordNet and fuzzy association rule mining for multi-label document clustering. *Data & Knowledge Engineering*, *69*(11), 1208–1226.

Chu, K.-M., & Li, F. (2010). Topic Evolution Based on LDA and Topic Association [J]. *Journal of Shanghai Jiaotong University*, *11*.

Clark, S. (2015). Vector Space Models of Lexical Meaning. *Handbook of Contemporary Semantics*, *10*, 9781118882139.

Clements, M., Vries, A. P. de, & Reinders, M. J. (2009). Exploiting Positive and Negative Graded Relevance Assessments for Content Recommendation. In *International Workshop on Algorithms and Models for the Web-Graph* (pp. 155–166).

Croft, W. B., Metzler, D., & Strohman, T. (2010). *Search Engines - Information Retrieval in Practice* (Vol. 520). Addison-Wesley Reading.

Cui, X., & Potok, T. E. (2005). Document clustering analysis based on hybrid PSO+ K-means algorithm. *Journal of Computer Sciences*, *27*, 33.

Deese, J., & Bechtel, W. (1990). Philosophy of Science: An Overview for Cognitive Science. *The American Journal of Psychology*, 122–124.

Diab, M., Hacioglu, K., & Jurafsky, D. (2004). Automatic tagging of Arabic text: From raw text to base phrase chunks. In *Proceedings of HLT-NAACL 2004: Short papers* (pp. 149–152).

Diederich, J., & Iofciu, T. (2006). Finding Communities of Practice from User Profiles Based On Folksonomies. *CEUR Workshop Proceedings*, *213*, 288–297.

Dietrich, D., et al.. (2015). *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data.* John Wiley & Sons,.

Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., & Zhang, F. (2017). A hybrid collaborative filtering model with deep structure for recommender systems. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017* (pp. 1309–1315).

Feldman, S. E. (1999). NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval. *23*, 62–73.

Fereday, J., & Muir-Cochrane, E. (2006). Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development. *International Journal of Qualitative Methods*, *5*(1), 80-92.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research*, *4*(Nov), 933–969.

Gemmis, M. de, Lops, P., Semeraro, G., & Musto, C. (2015). An investigation on the serendipity problem in recommender systems. *Information Processing Management*, *51*(5), 695 - 717.

Giles, J., Ang, K. K., Mihaylova, L. S., & Arvaneh, M. (2019). A Subject-to-subject Transfer Learning Framework Based on Jensen-Shannon Divergence for Improving Brain-computer Interface. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3087–3091).

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–71.

Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. *Synthesis Lectures on Human Language Technologies*, *10*(1), 1–309.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National academy of Sciences*, *101*(suppl 1), 5228–5235.

Hanani, U., Shapira, B., & Shoval, P. (2001). Information Filtering: Overview of Issues, Research and Systems. *User Modeling and User-Adapted Interaction*, *11*(3), 203–259.

Hassan, J., Tahir, M. A., & Ali, A. (2021). Natural language understanding of map navigation queries in Roman Urdu by joint entity and intent determination. *PeerJ Computer Science*, *7*, 15–20.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised Learning. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (pp. 485–585). Springer.

He, J., Chen, X., Du, M., & Jiang, H. (2015). Topic evolution analysis based on improved online LDA model. *Journal of Central South University (Science and Technology)*, 547-553.

Heylighen, F. (2002). Complexity and Information Overload in Society: why increasing efficiency leads to decreasing control.

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261–266.

Huang, A. (2008). Similarity measures for text document clustering. In *Proceedings of the 6th New Zealand Computer Science Research Student Conference* (Vol. 4, pp. 9–56).

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666.

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender Systems: An Introduction.* Cambridge University Press.

Jin, R., Si, L., & Zhai, C. (2002). Preference-based graphic models for collaborative filtering. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence* (pp. 329–336).

Jin, R., Si, L., Zhai, C., & Callan, J. (2003). Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of the twelfth International Conference on Information and Knowledge Management* (pp. 309–316).

Jin, Z., Cao, J., Zhang, Y., & Luo, J. (2016). News verification by exploiting conflicting social viewpoints in microblogs. *Proceedings of the AAAI Conference on Artificial Intelligence*, *30*(1).

Jones, K. S. (2004). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *60*, 493-502.

Katiyar, A., & Cardie, C. (2018). Nested Named Entity Recognition Revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 861–871). New Orleans, Louisiana: Association for Computational Linguistics.

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2017). Natural Language Processing: State of The Art, Current Trends and Challenges.

Koltcov, S., Ignatenko, V., Terpilovskii, M., & Rosso, P. (2021). Analysis and tuning of hierarchical topic models based on Renyi entropy approach. *PeerJ Computer Science.*

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised Machine Learning: A Review of Classification Techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies* (Vol. 160, pp. 3–24).

Krasnov, F. (2018). Topic Classification Through Topic Modeling with Additive Regularization for Collection of Scientific Papers. In *Proceedings of the 14th Central and Eastern European Software Engineering Conference Russia.* New York, NY, USA: Association for Computing Machinery.

Lau, J. H., Collier, N., & Baldwin, T. (2012). On-line Trend Analysis with Topic Models: twitter Trends Detection Topic Model Online. *Proceedings of COLING 2012*, 1519-1534.

Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, *41*(4, Part 2), 2065 - 2073.

Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook* (pp. 73–105). Springer.

Marsden, J. R., & Pingry, D. E. (2018). Numerical data quality in IS research and the implications for replication. *Decision Support Systems*, *115.*

Mateas, M., & Stern, A. (2004). Natural Language Understanding in Façade: Surface-Text Processing. In S. Göbel, U. Spierling, A. Hoffmann, I. Iurgel, O. Schneider, J. Dechau, & A. Feix (Eds.), *Technologies for Interactive Digital Storytelling and Entertainment* (pp. 3–13). Berlin, Heidelberg: Springer Berlin Heidelberg.

Mekonnen, A., & Abdullayev, S. (2017). Topic Modeling and Clustering for Analysis of Road Traffic Accidents. *Master of Science thesis. Department of Applied Mechanics, Chalmers University of Technology, Göteborg, Sweden*, 65.

Mimno, D., Hoffman, M., & Blei, D. (2012). Sparse Stochastic Inference for Latent Dirichlet allocation. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, 2.*

Mousavi, S. S., Schukat, M., & Howley, E. (2018). Deep Reinforcement Learning: An Overview. In Y. Bi, S. Kapoor, & R. Bhatia (Eds.), *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016* (pp. 426–440). Cham: Springer International Publishing.

Mugunthadevi, K., Punitha, S., Punithavalli, M., & Mugunthadevi, K. (2011). Survey on Feature Selection in Document Clustering. *International Journal on Computer Science and Engineering*, *3*(3), 1240–1241.

Mukherjee, M., & Poovammal, E. (2019). Improved topic modeling with parallel-supervised LDA. *International Journal of Recent Technology and Engineering.*

Naak, A., Hage, H., & Aimeur, E. (2009). A Multi-criteria Collaborative Filtering Approach for Research Paper Recommendation in Papyres. In *E-Technologies: Innovation in an Open World* (pp. 25–39).

Nadeau, D., & Sekine, S. (2007). A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*, *30*(1), 3–26.

Nasukawa, T., & Yi, J. (2003). Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd International Conference on Knowledge Capture* (pp. 70–77).

Nation, K., Snowling, M. J., & Clarke, P. (2007). Dissecting the relationship between language skills and learning to read: Semantic and phonological contributions to new vocabulary learning in children with poor reading comprehension. *International Journal of Speech-Language Pathology*, *9*(2), 131–139.

Newman, D., Lau, J. H., Grieser, K., & Baldwin, T. (2010). Automatic Evaluation of Topic Coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL* (pp. 100–108).

Newman, D., Noh, Y., Talley, E., Karimi, S., & Baldwin, T. (2010). Evaluating Topic Models for Digital Libraries. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries* (pp. 215–224).

Nilashi, M., Jannach, D., Ibrahim, O. bin, Esfahani, M. D., & Ahmadi, H. (2016). Recommendation quality, transparency, and website quality for trust-building in recommendation agents. *Electronic Commerce Research and Applications.*

Olivier, M. S. (2009). *Information technology research: A practical guide for computer science and informatics.* Van Schaik.

Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications*, *39*(11), 10059–10072.

Pavlov, D., Manavoglu, E., Giles, C. L., & Pennock, D. M. (2004). Collaborative filtering with maximum entropy. *IEEE Intelligent Systems*, *19*(6), 40–47.

Peng, Y., Kou, G., Chen, Z., & Shi, Y. (2006). Recent trends in Data Mining (DM): Document Clustering of DM Publications. In *2006 International Conference on Service Systems and Service Management* (Vol. 2, pp. 1653–1659).

Porter, M. F. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, *14*(3), 130-137.

Purpura, A. (2018). Non-negative matrix factorization for topic modeling. In *Ceur workshop proceedings.*

Qomariyah, S., Iriawan, N., & Fithriasari, K. (2019). Topic modeling Twitter data using Latent Dirichlet Allocation and Latent Semantic Analysis. In *Aip conference proceedings.*

Ramirez, E. H., Brena, R., Magatti, D., & Stella, F. (2012). Topic model validation. *Neurocomputing*, *76*(1), 125–133.

Ranjan, P., Ray, Harish, V., Sarkar, S., & Basu, A. (2003). Part of Speech Tagging and Local Word Grouping Techniques for Natural Language

Parsing in Hindi. In *Proceedings of the 1st International Conference on Natural Language Processing (ICON 2003).*

Rehurek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks* (p. 45-50).

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work* (pp. 175–186).

Resnick, P., & Varian, H. R. (1997). Recommender Systems. *Communications of the ACM*, *40*(3), 56–58.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook* (pp. 1–35). Springer.

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,.

Sajjadi, M. S. M., Bachem, O., Lucic, M., Bousquet, O., & Gelly, S. (2018). Assessing Generative Models via Precision and Recall. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (p. 5234–5243).

Sang, E. F., & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, 127–132.

Santana, C. (2016). What is language? *Ergo, an Open Access Journal of Philosophy, 3.*

Saunders, M., Lewis, P., Thornhill, A., & Bristow, A. (2019). ”Research Methods for Business Students” Chapter 4: Understanding research philosophy and approaches to theory development. *Research Methods for Business Students*, 128-171.

Savolainen, R. (2015). Cognitive barriers to information seeking: A conceptual analysis. *Journal of Information Science*, *41*(5), 613–623.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, *34*(1), 1–47.

Shah, K., Salunke, A., Dongare, S., & Antala, K. (2017). Recommender systems: An overview of different approaches to recommendations. In *2017 international conference on innovations in information, embedded and communication systems (iciiecs)* (p. 1-4).

Shah, N., & Mahajan, S. (2012). Document clustering: a detailed review. *International Journal of Applied Information Systems*, *4*(5), 30–38.

Shardanand, U., & Maes, P. (1995). Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vol. 95, pp. 210–217).

Sharma, R., & Singh, R. (2016). Evolution of Recommender Systems from Ancient Times to Modern Era: A Survey. *Indian Journal of Science and Technology*, *9*(20), 1–12.

Singh, P. (2019). Natural language processing. In *Machine learning with pyspark* (pp. 191–218). Springer.

Steenkamp, A., & Mccord, A. (2007). Approach to Teaching Research Methodology for Information Technology. *Journal of Information Systems Education*.

Stevens, K., Kegelmeyer, P., Andrzejewski, D., & Buttler, D. (2012). Exploring Topic Coherence over Many Models and Many Topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 952–961). Jeju Island, Korea: Association for Computational Linguistics.

Storey, V. C., & Song, I.-Y. (2017). Big data technologies and Management: What conceptual modeling can do. *Data & Knowledge Engineering*, *108*, 50–67.

Suhaimi, F., & Hussin, N. (2017). The Influence of Information Overload on Students' Academic Performance. *International Journal of Academic Research in Business and Social Sciences*, *7*.

Tapaswi, N., & Jain, S. (2012). Treebank based deep grammar acquisition and Part-Of-Speech Tagging for Sanskrit sentences. In *2012 CSI Sixth International Conference on Software Engineering (CONSEG)* (pp. 1–4).

Tong, Z., & Zhang, H. (2016). A Text Mining Research Based on LDA Topic Modelling. In *Computer Science Information Technology* (pp. 201–210).

Uto, M., Louvigné, S., Kato, Y., Ishii, T., & Miyazawa, Y. (2017). Diverse reports recommendation system based on Latent Dirichlet Allocation. *Behaviormetrika*, *44*(2), 425–444.

Wallach, H., Mimno, D., & McCallum, A. (2009). Rethinking LDA: Why Priors Matter. In *Advances in Neural Information Processing Systems* (Vol. 22).

Walton, D. (2009). Argumentation theory: A very short introduction. In *Argumentation in Artificial Intelligence* (pp. 1–22). Springer.

Wei, C.-P., Yang, C.-S., Hsiao, H.-W., & Cheng, T.-H. (2006). Combining preference-and content-based approaches for improving document clustering effectiveness. *Information Processing & Management*, *42*(2), 350–372.

Wei, X., & Croft, W. B. (2006). LDA-Based Document Models for Ad-Hoc Retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 178–185).

Willett, P. (1988). Recent trends in hierarchic document clustering: A critical review. *Information Processing & Management*, *24*(5), 577–597.

William, C., Robert, E. S., & Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research*, *10*, 243–270.

Wilson, J. (2014). *Essentials of business research: A guide to doing your research project.* Sage.

Yeh, J.-h., & Wu, M.-l. (2010). Recommendation Based on Latent Topics and Social Network Analysis. In *2010 Second International Conference on Computer Engineering and Applications* (Vol. 1, p. 209-213).

Yi, J., Nasukawa, T., Bunescu, R., & Niblack, W. (2003). Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings - IEEE International Conference on Data Mining, ICDM* (pp. 427–434).

Yi, X., & Allan, J. (2009). A Comparative Study of Utilizing Topic Models for Information Retrieval. In *Advances in Information Retrieval* (pp. 29–41).

Appendix A

Expansion of validation process

The validation process described in Section 7.5, focused on providing observations based on the demonstration of the prototype and model of two topics. Even though Section 7.5 only showcased two topics, more tests were conducted which will be displayed here.

The layout of the sections will include one topic at which the top 10 recommendations were presented.

A.1 Topic: Cloud Computing

The title of the test document: 'The Management of Security in Cloud Computing'.

TITLE	**JSD SCORE**
Secure e-Government Services: Towards A Framework	0.208068
Android Botnets on the Rise: Trends and Characteristics	0.236715
Protection of personal information in the South-African	0.296202
Security Foundation for a Distributed Cabin Core	0.300801
Considering web services security policy compatibility	0.315992
Cloud Supply Chain Resilience- A Coordination	0.316091
Security Objectives, Controls and Metrics Development	0.322413
Evaluating Information Security Controls Applied	0.335019
Considering the influence of human trust in practical	0.33537
A Software Gateway to Affordable and Effective	0.342548

Table A.1: Security in Cloud Computing similarity

A.2 Topic: Neural Network

The title of the test document: ‘Filtering Spam E-Mail with Generalized Additive Neural Networks’.

TITLE	JSD SCORE
Investigating the effect of genetic algorithms	0.288162
Spam over Internet Telephony and how to deal with it	0.309555
E-mail security awareness at Nelson Mandela	0.320261
An Analysis of authentication for passive RFID	0.32099
The use of computer vision technologies	0.340537
The IP Protection of Electrical Databases	0.343205
Remote Fingerprinting and Multisensor Data Fusion	0.343431
AFA-RFID-Physical Layer Authentication for	0.350931
Mobile Security from an Information Warfare	0.351308
Bimodal Biometrics for Financial Infrastructure	0.352621

Table A.2: Neural Network similarity

A.3 Topic: Smartphone Security Awareness

The title of the test document: ‘Exploring End-User Smartphone Security Awareness within a South African context’.

TITLE	JSD SCORE
An adaptation of the awareness boundary model	0.213688
Identity Management for e-Government Libya	0.242323
Considering the influence of human trust	0.247079
Factors Affecting User Experience with Security	0.252316
Online Social Networks: Enhancing user trust	0.272419
Gamifying Authentication	0.30036
Secure Cloud Computing Benefits, Risks	0.300522
Towards achieving scalability and interoperability	0.300544
A formal qualitative risk management approach	0.301527
Design of cyber security awareness game	0.309103

Table A.3: Awareness similarity

A.4 Topic: Bring Your Own Device

The title of the test document: ‘A framework towards governing Bring Your Own Device in SMMEs’.

TITLE	JSD SCORE
Considering the influence of human trust	0.207733
Protection of personal information in the South African	0.250983
Information Security Assurance Model (ISAM)	0.256613
Bloom’S Taxonomy for Information Security Education	0.257684
A survey of computer crime and security in South Africa	0.258457
The impact of Information security awareness training	0.259873
The Identification of Information Sources to	0.265763
BC3I – Towards requirements specification	0.267988
A formal qualitative risk management approach	0.271136
An Interactive Visual Library Model to Improve	0.271961

Table A.4: BYOD similarity

A.5 Topic: Intrusion Detection

The title of the test document: ‘Intrusion Detection in Bluetooth Enabled Mobile Phones’.

TITLE	JSD SCORE
A Framework of Opportunity-Reducing Techniques	0.166268
A Conceptual Opportunity-based Framework	0.198728
The Murky Waters of IT Governance	0.269619
Towards a framework for a network warfare capability	0.282355
Combatting Phishing: A Holistic Human Approach	0.299802
Towards a PHP Webshell Taxonomy using Deobfuscation	0.30253
Mobile Security from an Information Warfare Perspective	0.309996
Secure e-Government Services: Towards A Framework	0.317646
Unsolicited Short Message Service Marketing	0.318187
Towards a Sandbox for the Deobfuscation	0.32363

Table A.5: Intrusion detection similarity