

|  |
| --- |
| School of Computing  Faculty of Engineering AND PHYSICAL SCIENCES |

Supervisor Recommendation System

Jianmeng Tian

Submitted in accordance with the requirements for the degree of  
MSc Advance Computer Science

**2022/2023**

The candidate confirms that the following have been submitted*:*

*<As an example>*

|  |  |  |
| --- | --- | --- |
| **Items** | **Format** | **Recipient(s) and Date** |
| *Deliverables 1* | *Report* | *SSO (18/08/2023)* |
| *Deliverable 2* | *Project codes, dataset and outcomes* | *Supervisor, assessor (18/08/2023)* |

Type of Project: Exploratory software

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)

© 2023 The University of Leeds and Jianmeng Tian

# Summary

This project proposes an approach to help students in computing school of the University of Leeds find suitable supervisors for their final projects. Leveraging multiple machine learning and NLP techniques, this project offers a system that recommends supervisors based on user input, generates summaries for each potential supervisor and recommend similar supervisors to users. The methodologies encompassed include various vectorization techniques, recommendation algorithms, text generation models, topic modelling, and clustering recommendation. Using the OpenAI API for vector generation and text generation further enhanced the system's performance. The study illustrates the strengths and weaknesses of each approach and indicates the most effective techniques for the specific context of the project.

# Table of Contents

Summary iii

Table of Contents iv

Chapter 1 Introduction 1

1.1 Problem statement 1

1.2 Project Aim 1

1.3 Objectives 2

1.4 Deliverables 3

Chapter 2 Background Research 4

2.1 Literature Review 4

2.1.1 Vectorization Methods 4

2.1.2 Recommendation Systems 5

2.1.3 Text Generation Models 6

2.1.4 Topic Modelling 6

2.1.5 OpenAI API 6

2.1.6 Clustering Recommendation 7

2.2 Methods and Techniques 7

2.3 Choice of methods 8

Chapter 3 System Design and Experiments Design 10

3.1 Technical Requirements 10

3.2 System Design 11

3.2.1 System Overview 11

3.2.2 System Components 12

3.2.3 Future Scope 12

3.3 Experiments Design 12

3.3.1 Data Scraping 12

3.3.2 Data Cleaning 13

3.3.3 Recommendation Algorithms Analysis 13

3.3.4 Vectorization Techniques Analysis 13

3.3.5 Text Generation Tools Analysis 13

3.3.6 Topic Modelling Techniques Analysis 13

3.3.7 OpenAI Techniques Integration 13

3.3.8 Clustering Recommendation Analysis 14

Chapter 4 Implementation 15

4.1 Web Scraping 15

4.2 Data Cleaning 15

4.3 Recommendation Algorithms Analysis 16

4.3.1 Jaccard Similarity Analysis 16

4.3.2 Cosine Similarity Analysis 16

4.3.3 Comparison and Results 17

4.3.3.1 Experiment 1 17

4.3.3.1 Experiment 2 18

4.4 Vectorization Techniques Analysis 19

4.4.1 Bag-of-Words Models 19

4.4.1.1 CountVectorizer 19

4.4.1.2 TF-IDF 21

4.4.2 Word Embedding Models 22

4.4.2.1 Word2Vec 22

4.4.2.2 Doc2Vec 22

4.4.3 Experiments and Results 23

4.5 Text Generation Tools Analysis 24

4.6 Topic Modelling Techniques Analysis 25

4.6.1 LDA and NMF 26

4.6.2 BERTopic 26

4.6.3 Conclusion 27

4.7 OpenAI Techniques Integration 27

4.7.1 text-davinci-003 27

4.7.2 text-embedding-ada-002 27

4.8 Clustering Recommendation Analysis 28

4.8.1 Data Pre-processing 28

4.8.2 Clustering Analysis 29

4.8.3 Supervisor Recommendation 30

Chapter 5 Evaluation 31

5.1 Recommendation Approaches 31

5.2 Vectorization Methods 31

5.2.1 Bag of Words models 31

5.2.2 Embedding models 32

5.2.3 Summary 32

5.3 Topic Modelling Techniques 32

5.4 Clustering Recommendation Analysis 32

5.5 Summary 33

5.6 Hypothetical Use Cases 33

5.6.1 Use case 1 33

5.6.2 Use case 2 34

5.6.3 Use case 3 35

Chapter 6 Conclusion & Future Work 37

6.1 Conclusion 37

6.2 Future Work 38

List of References 40

Appendix A External Materials 43

A.1 GitHub Repository of the Project 43

A.2 Dataset acquisition source 43

A.3 External Libraries 43

A.4 External codes (silhouette analysis) learned from scikit-learn.org 44

Appendix B Ethical Issues Addressed 45

B.1 Privacy and Consent 45

B.2 Data Use 45

B.3 Equity and Bias 45

# Chapter 1 Introduction

## 1.1 Problem statement

The process of finding an appropriate supervisor of their final projects can be confusing and time-consuming for students. Currently, students have to read through supervisor profiles one by one, and given that the School of Computing now has approximate 100 supervisors, this process takes at least an hour. If the number of supervisors is increased in the future, they will spend more time on selecting supervisors. Moreover, the process of selecting supervisors was complex. After reading through a large number of profiles, students also need to compare and rank supervisors manually, then they can submit their rank of 6 selected supervisors. As a result, the current supervisor selection process is inefficient and results in many students submitted supervisors who do not match their academic background or the needs of their proposed project. Therefore, the necessity of building an effective system that can alleviate this burden and optimize the supervisor selection process is significant.

## 1.2 Project Aim

This project aims to address this issue by investigating and evaluating various machine learning and natural language processing approaches to determine the most optimal approach for developing a supervisor recommendation system. Through massive experimentation and analysis, the project seeks to identify the most effective recommendation approaches, vectorization techniques, and text generation models that can effectively address the requirements of a supervisor recommendation system.

By conducting a thorough analysis of different recommendation approaches, such as Jaccard similarity and cosine similarity, then use cosine similarity along with various vectorization methods like Word2Vec, Doc2Vec, CountVectorizer, and TF-IDF, this project will assess their respective performances and suitability in the context of the supervisor recommendation task. Furthermore, the project will explore text generation models, such as T5, Bart, and TextRank, to provide concise and informative summaries of each supervisor to students.

Another part of the project is the investigation of topic modelling techniques, including LDA, NMF, and BERTopic, to determine their potential for recommendation and compare with classic recommendation approaches. By utilizing these techniques, the system can find latent topics and identify supervisors with similar backgrounds to recommend supervisors have same topics with student’s demand.

Additionally, the project will leverage the capabilities of OpenAI APIs, specifically text-davinci-003 for text generation and text-embedding-ada-002 for vector generation, aligning with classic recommendation approaches to augment the recommendation system's performance and accuracy. The project will assess the capabilities of these APIs, exploring their potential to enhance the quality of generated topics and embeddings.

At last, this project tested and evaluated clustering recommendation based on KMeans clustering. The system will allocate supervisors into different clusters based on their background and research area, then if the user has already selected certain supervisors, cluster recommendation will recommend similar supervisors to user. This approach includes standardization, PCA dimension reduction and silhouette analysis techniques to enhance the final clusters outcome and performance of recommendation.

Overall, by conducting extensive experiments and evaluations, this project aims to provide valuable insights into the effectiveness of various vectorization, recommendation, topic modelling approaches and clustering recommendation for developing a supervisor recommendation system. The findings and conclusions of this project will contribute to the broader field of recommendation systems and assist in the future development of efficient and accurate recommendation mechanisms designed specifically to the supervisor selection system in academic institutions.

## 1.3 Objectives

1. Obtain personal profiles of supervisors in the School of Computing from the website of University of Leeds by Using a web scraping approach, beautiful soup (Richardson, 2007).

2. Clean the dataset: expand abbreviations, remove generic nouns, remove html expressions and special characters. Prepare another dataset based on the former but without subjects and connectives.

3. To analyse and compare different recommendation algorithms, cosine similarity and Jaccard similarity, in order to identify the most effective one for recommendation system.

4. Evaluate various vectorization techniques such as Word2Vec, Doc2Vec, CountVectorizer, and TF-IDF for their efficiency and applicability to the dataset and recommendation algorithm.

5. Generate brief introductions for each supervisors using text generation models like T5, Bart, or TextRank.

6. Explore topic modelling techniques (LDA, NMF, BERTopic) and OpenAI APIs (text-davinci-003 and text-embedding-ada-002) to compare with classic recommendation approaches.

7. Design a cluster-based recommendation approach that recommends supervisors with similar research topics or backgrounds.

## 1.4 Deliverables

1. A cleaned and processed dataset of faculty members in the School of Computing at the University of Leeds, including their profile, Areas of expertise, and published works. (Objectives 1, 2)

2. A comparative study of different vectorization techniques and recommendation methods, along with their strengths, weaknesses, and the rationale for the selected method. (Objectives 3, 4)

3. Providing a brief introduction for each supervisor. (Objectives 5)

4. A comprehensive evaluation of different topic modelling techniques and OpenAI APIs, compare with the former tested techniques and justify the final chosen one. (Objectives 6)

6. Implementation of a clustering recommendation system, providing clusters of similar supervisors to students based on their inputs. (Objectives 6)

5. A final recommendation system that takes user input and outputs a list of recommended supervisors and their summaries. (Objectives 7)

7. A GitHub repository that contains the source code of experiments and the recommendation system.

8. The MSc project report.

# Chapter 2 Background Research

This chapter will review various literature across multiple research fields, focusing on vectorization methods, recommendation systems, text generation models, topic modeling, the OpenAI API, and clustering recommendation. These methodologies provide the basis for the project, which aims to create an effective system for recommending supervisors to students. This chapter outlines the advantages and limitations of each approach and discusses the potential for their application in the context of this project.

## 2.1 Literature Review

The project is grounded in a variety of research fields extensively investigated in the literature, each contributing valuable methodologies and insights to the proposed system.

### 2.1.1 Vectorization Methods

Vectorization is a crucial process in Natural Language Processing (NLP), transforming textual data into numerical vectors that machine learning algorithms can interpret and analyze. The cosine similarity approach mentioned above would require vectorised text for the calculation. Various vectorization techniques have been proposed and used over the years, each with unique properties and advantages. Two types of vectorisation techniques are widely used nowadays and considered applicable to this project, the first is Bag of Words (BoW) models:

CountVectorizer, for instance, offers a simple method to tokenize a collection of text documents and build a vocabulary of known words, but it treats each word as a discrete entity, ignoring any semantic relationship between words (scikit-learn developers, 2021). It will convert the text into numerical data, forming a matrix where each row represents a document (sentence) and each column represents a word.

On the other hand, TF-IDF, or Term Frequency-Inverse Document Frequency, is a statistical method that reflects how important a word is to a document in a collection (Ramos, 2003). Unlike CountVectorizer, TF-IDF gives more importance to words that are less frequent in the corpus, reducing the importance of common words that are less informative.

Another is word embedding models:

Word2Vec and Doc2Vec are prediction-based methods that go beyond simple tokenization and frequency counts. These techniques learn continuous word representations using shallow neural networks, capturing semantic relationships between words and even entire documents (Mikolov et al., 2013) (Le & Mikolov, 2014). Word2Vec is trained on individual words, while Doc2Vec is a model that represents each document or sentence as a vector.

### 2.1.2 Recommendation Systems

Recommendation systems have become a critical component in numerous applications related to information retrieval and personalized user services. The core principle behind recommendation systems is to provide users with personalized suggestions based on their preferences and behaviour. In the context of this project, the system will take students as users and find the right supervisors for them based on their needs.

Various approaches are utilized in recommendation systems, including Collaborative Filtering, Content-Based approach, Hybrid approach, Social approach, and Demographic Approach (Al Fararni et al., 2020). Collaborative Filtering fulfils the needs of this project. It usually uses for analysing historical user data (like purchase and browsing history) to identify similar users or items, then recommends based on these similarities. In this project, it uses for analysis supervisors’ data and recommend them for students.

Two common techniques used in Collaborative Filtering systems are cosine similarity and Jaccard similarity. They usually serve as measures to calculate the similarity between different items in recommendation system, thus aiding in suggesting the most appropriate items to users.

Cosine similarity, a measure of similarity between two non-zero vectors in an inner product space, gauges the cosine of the angle between these vectors. Cosine similarity is particularly beneficial when managing high-dimensional data, like text data (Verma and Aggarwal, 2020) (Huang et al., 2021). The text data in documents firstly covert to vectors which represents word frequency or TF-IDF weight. Then the cosine similarity can use them to calculate similarity of different documents.

On the other hand, Jaccard similarity measures the similarity between two nominal attributes by calculating the size of the intersection divided by the size of the union of the two sets. It has been shown to be particularly effective when applied to binary or discrete dataset (Verma and Aggarwal, 2020). For example, Jaccard similarity can be used to compare the similarity between two gene samples (Huang et al., 2021).

In conclusion, both cosine similarity and Jaccard similarity have their strengths in different scenarios and data types, and the choice of method depends on the specific application scenario and requirements (Verma and Aggarwal, 2020). These methods play a pivotal role in constructing Collaborative Filtering recommendation systems.

### 2.1.3 Text Generation Models

Text generation models and automatic summarization methods have gained considerable attention in NLP research. These models provide a way to generate human-like text, which is a critical requirement for the proposed system.

TextRank, an extractive summarization technique, brings order into text by ranking the relevance and significance of sentences in a document, similar to how PageRank ranks web pages (Mihalcea & Tarau, 2004). Transformer-based models like T5 (Raffel et al., 2019) and Bart (Lewis et al., 2019) have achieved state-of-the-art performance on language generation tasks (Ao et al., 2021). These models adopt the transformer architecture (Vaswani et al., 2017) which is based on self-attention mechanisms, allowing the models to generate more coherent and contextually relevant sentences.

### 2.1.4 Topic Modelling

Topic modelling techniques can discover abstract topics from documents.

Latent Dirichlet Allocation (LDA) is a method to find topics in texts using a probabilistic model (Blei et al., 2003). It assumes each text has a mixture of topics, and each topic has a distribution of words. It infers the topics and their weights in each text from the words.

Non-negative Matrix Factorization (NMF) is a method to find topics in texts by decomposing a matrix into two smaller matrices (Lee & Seung, 1999), where the matrix represents the frequency of words in each text, and the smaller matrices represent the topics and their weights in each text, and the words and their weights in each topic.

BERTopic is a novel topic modelling technique that combines BERT embeddings (Devlin et al., 2018) and class-based variation of TF-IDF. It generates document embeddings with pre-trained transformer-based language models, clusters these embeddings, and then generates topic representations with a class-based variation of TF-IDF (Grootendorst, 2020).

### 2.1.5 OpenAI API

The rise of large transformer-based architectures has dramatically reshaped the NLP area, and widely used commercial models such as ChatGPT and Claude have proved their strength to the public. OpenAI's text-davinci-003 and text-embedding-ada-002 are two such examples that have been developed to allow easy access to these powerful models (OpenAI, 2021). The former is capable of generating human-like text based on a prompt, while the latter produces embeddings for the given input text, which can be beneficial for downstream tasks like clustering or similarity estimation.

### 2.1.6 Clustering Recommendation

Clustering algorithms, such as KMeans, have been employed in various applications for grouping similar instances together. KMeans works well with high dimensional data, especially when dimension reduction techniques like Principal Component Analysis (PCA) are used to capture the essential structure of the data (Jolliffe & Cadima, 2016). Before applying KMeans, it's often advisable to standardize the data to ensure all features have the same scale (Hastie et.al, 2009). To determine the optimal number of clusters, silhouette analysis can be used, which provides a succinct graphical representation of how well each object lies within its cluster (Rousseeuw, 1987).

## 2.2 Methods and Techniques

The project will utilize a range of methods and techniques derived from the aforementioned areas of research. The choice of these methods is guided by their relevance to the problem at hand and their performance in respective contexts as suggested by the literature.

In Vectorization, various methods such as CountVectorizer, Word2Vec, Doc2Vec, TF-IDF and OpenAI’s text-embedding-ada-002 will be evaluated for their efficacy and compatibility with the dataset and the recommendation algorithm. These methods have been chosen as they offer a balance between computational efficiency and ability to effectively capture the semantics of the text data (scikit-learn developers, 2021) (Mikolov et al., 2013) (Le & Mikolov, 2014) (Ramos, 2003) (OpenAI, 2021).

For recommender system, as this system needs to make recommendation based on text content, and cosine similarity and Jaccard similarity has ability to effectively compute similarity between different text content, which is vital for the recommendation task (Adomavicius & Tuzhilin, 2005). So, they will be evaluated separately and their performance and compatibility will be compared. The cosine similarity will be evaluated using different vectorisation methods as mentioned above.

Text generation for brief introductions of each supervisor will be carried out using models like T5, Bart, or TextRank. The selection of these models is based on their ability to generate high-quality, human-like text and their performance in recent NLP research (Mihalcea & Tarau, 2004) (Raffel et al., 2019) (Lewis et al., 2019).

Topic modeling techniques including LDA, NMF, BERTopic will be explored. These techniques, known for their effectiveness in discovering latent topics and generating high-quality embeddings respectively, are chosen to compare their performance with classic vector-calculation-based recommendation approaches (Blei et al., 2003) (Grootendorst, 2020) (Devlin et al., 2018).

Finally, a cluster-based recommendation approach will be designed that leverages KMeans clustering, standardization, PCA dimension reduction, and silhouette analysis techniques. The selection of these methods is based on their proven effectiveness in grouping similar instances together and handling high dimensional data (Jolliffe & Cadima, 2016) (Hastie et.al, 2009) (Rousseeuw, 1987).

## 2.3 Choice of methods

The selection of methods for this project is driven by the specific requirements and objectives of the supervisor recommendation task faced to students. The chosen methods aim to optimize the supervisor selection process by reducing the time and improving the final accuracy for matching teachers to students compared to manual selection.

Therefore, the first step is selecting the most optimal recommendation algorithm from cosine similarity and Jaccard similarity, based on their performance and scalability. This is followed by testing to select the most suitable vectorization methods for various recommendation algorithms and other techniques. All these methods are analyzed considering their performance and matching in two specific contexts: supervisor information text and user input text.

Subsequently, the potential of topic modelling recommendation methods is investigated. These methods can allocate the topics for supervisors and users based on their contents and inputs, and match topics for users. From LDA, NMF, and BERTopic, the highest performing ones will be chosen, compared with vector-calculation-based recommendation methods, and then analysed to conclude which recommendation approach is most effective for this system.

Inspired by the topic modelling approaches, the project will also assess the performance of OpenAI's text-davinci-003 for generating topics from supervisors' contents. These topics, along with vector-based approaches, will be utilized to recommend supervisors to users. OpenAI's text-embedding-ada-002 will be used for vectorising the supervisors’ contents and users’ inputs, and compare with the traditional vectorisation approaches. Exploring the applicability and performance of OpenAI's API and finally summarising the best performing recommendation approaches and vectorisation approaches for this system.

Lastly, due to their potential to identify latent topics, cluster similar supervisors, and recommend supervisors with similar research interests or backgrounds to students, clustering recommendation approaches were selected for testing. KMeans clustering will be implemented, with standardization and PCA dimension reduction employed as data pre-processing tools. Silhouette analysis will be used to find the most appropriate number of clusters.

The selection of these approaches is also considered by the need for the system to be scalable, economic efficiency and capable of handling a growing number of supervisors and different background of supervisors in the future.

# Chapter 3 System Design and Experiments Design

This chapter outlines the design and experimental approaches for a supervisor recommendation system. The system, implemented in a Jupyter notebook, allows students (users) to find suitable supervisors based on their input text. It includes components for supervisor recommendation, information summarization, and similar supervisor suggestion. While the current system is a proof-of-concept, plans for future development include a more interactive interface with additional features. The chapter also describes the design of experiments involving data scraping and cleaning, analysis of recommendation algorithms and vectorization techniques, assessment of text generation tools and topic modeling techniques, integration of OpenAI technologies, and testing a clustering recommendation approach. These experiments are designed to evaluate and optimize the effectiveness of the recommendation system.

## 3.1 Technical Requirements

As this project implemented in a Jupyter notebook, it requires an operating system that can run Python 3.7 or higher version, and libraries including BeautifulSoup, pandas, NumPy, scikit-learn, gensim, NLTK and transformers. It also necessitates a Jupyter Notebook environment for execution, an OpenAI API key and an available bank account linked to OpenAI. Hardware requirements include a chip with equivalent computing power to Apple M1 chip, at least 8GB RAM and network connection.

Based on these hardware and software requirements, the system should be able to meet the following requirements:

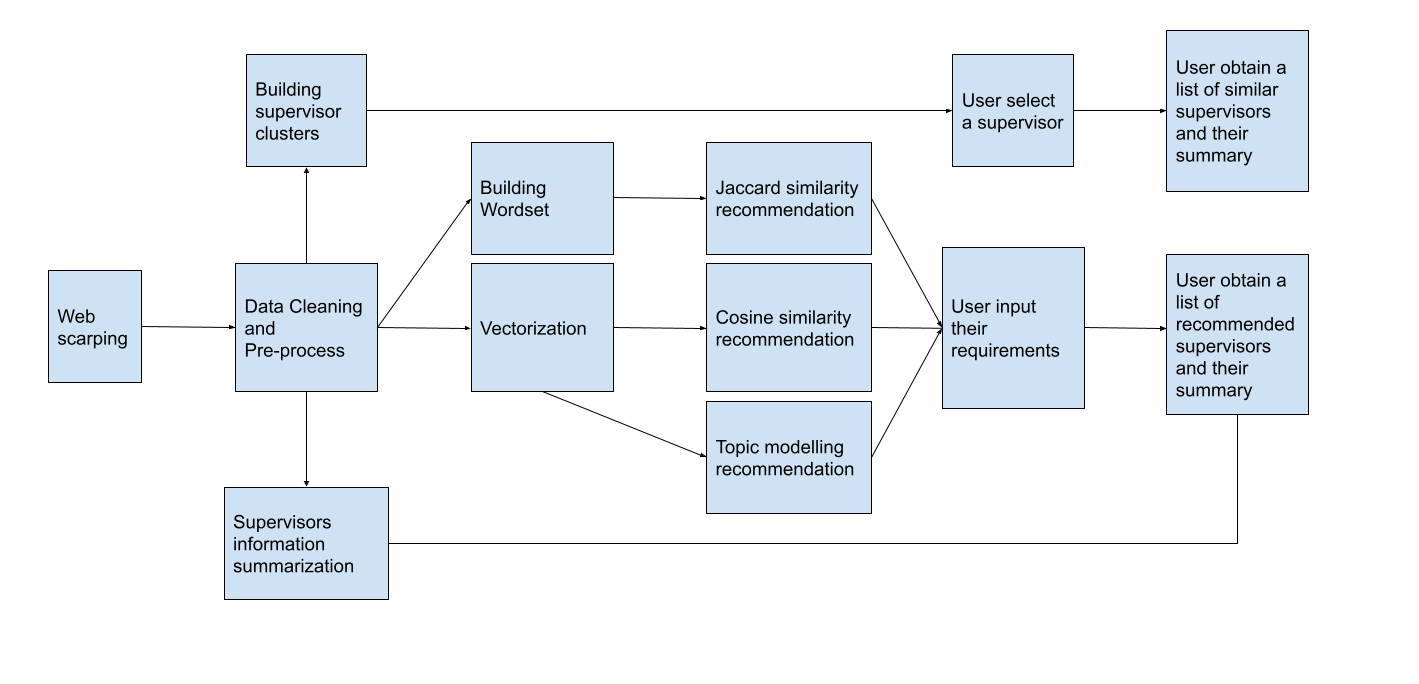
* The system should provide supervisor information dataset that are cleared and readable that can use in recommender systems.
* The system should provide informative introduction of supervisors based on the dataset.
* The system should produce a recommendation of supervisors based on the user input.
* The system should import the cleaned dataset and produce clusters of supervisors.
* The system should produce a recommendation of similar supervisors based on the cluster classification.

## 3.2 System Design

The core objective of this project is identification and exploration of applicable technologies and approaches for a supervisor recommendation system intended for students (users). While the project's emphasis is not on system design itself, it is focused on the techniques and approaches which support the system. As part of this project, a user-oriented recommendation system has been implemented within a Jupyter notebook.

## 3.2.1 System Overview

This system is designed to streamline the supervisor selection process for students. It has the capability to analyse user input text of any length and generate a list of suitable supervisors, sorted by match score in descending order. Additionally, the system provides a summary of each supervisor's information, sparing users the need to visit every supervisor’s individual web page. The system can also offer suggestions of similar supervisors based on a selected supervisor, assisting users in uncovering more potential matches. The following flowchart will briefly show how the system was built:



**Figures 3.2.1 Flowchart**

This flowchart shows the core functions (components) of the system and the process of building them.

## 3.2.2 System Components

Supervisor Recommendation Function: Users provide texts include their background or project requirements. Based on user input, the system generates a sorted list of potential supervisors. These supervisors are ranked according to their match score, calculated using the methods identified in the research.

Supervisor Information Summarization Function: For every recommended supervisor, the system generates a summary of their information. This summary includes key aspects such as their Curriculum Vitae, research interests, history projects and current students’ projects, also even includes personal hobbies, offering users a quick overview without the need to visit the supervisors' individual web pages.

Similar Supervisors Recommendation Function: If a user selects a supervisor, the system provides a cluster of other supervisors with common ground. This feature enables users to explore alternative options that align with their preferences or gives them a chance to fill their selected supervisor list.

## 3.2.3 Future Scope

The current system, implemented within a Jupyter notebook, serves as a proof-of-concept for the underlying technologies and methodologies. It lays the groundwork for a fully functional, user-oriented supervisor recommendation system. This future system could offer a more interactive web front-end interface, with additional features like user accounts, saved preferences, and more.

## 3.3 Experiments Design

The experiments design of this project comprises in the following stages.

## 3.3.1 Data Scraping

The primary data source for this project is the 'People' page of the School of Computing in University of Leeds. Information about prospective supervisors is scraped from individual web pages using the Beautiful Soup library. This information, which is the foundation of this project, includes the supervisors' professional profiles, their research interests, past and current projects, and even their hobbies.

## 3.3.2 Data Cleaning

Initial data cleaning involves removing HTML tags, special characters, and common words such as 'publications', 'Research projects', and 'Profile'. Abbreviations are expanded to their full forms to ensure consistency. An additional cleaning strategy has been designed to eliminate all subject pronouns and conjunctions, such as 'I' and 'and', to minimize their influence during text vectorization. The effectiveness of these cleaning strategies is evaluated by comparing the results before and after cleaning.

## 3.3.3 Recommendation Algorithms Analysis

Two recommendation algorithms, cosine similarity and Jaccard similarity will be analysed. The performance and results of these algorithms are compared to determine their suitability for this project. They will be compared using the same user input.

## 3.3.4 Vectorization Techniques Analysis

The cosine similarity algorithm is then compared with several vectorization techniques, namely Word2Vec, Doc2Vec, CountVectorizer, and TF-IDF. The performance and results of these techniques are analysed under the same conditions to investigate their effectiveness in this context.

## 3.3.5 Text Generation Tools Analysis

Different text generation tools (T5, Bart, TextRank) will be assessed. They are all used to generate a summary of information for each supervisor. Their performance will be examined in terms of speed of generating, length of content, readability and whether they cover full range of all points.

## 3.3.6 Topic Modelling Techniques Analysis

Several topic modelling techniques (LDA, NMF, BERTopic) will be tested. The results and performance of these techniques are contrasted with the traditional cosine similarity algorithm under same user input to identify their performance.

## 3.3.7 OpenAI Techniques Integration

OpenAI technologies will be tested and integrated into the project. First, the text generation API (text-davinci-003) is used to generate topic for each supervisor, then topics will be vectorized and fed into cosine similarity algorithm for recommendation. Second, the embedding API (text-embedding-ada-002) is used to directly vectorize all texts for recommendation using cosine similarity. The performance and cost-effectiveness of OpenAI techniques will be compared with traditional approaches.

## 3.3.8 Clustering Recommendation Analysis

In the final stage, a clustering recommendation technique will be tested. The vectorized text will be pre-processed using standardization and PCA dimension reduction. The silhouette analysis is performed with graphs, and the silhouette score is printed to determine the optimal number of clusters. The optimal number of clusters will be used for KMeans clustering of all supervisors. The effectiveness and rationality of supervisor classification will be tested.

# Chapter 4 Implementation

## 4.1 Web Scraping

The web scraping process was carried out using the BeautifulSoup library in Python, scraping from the 'People' page of the School of Computing at the University of Leeds (<https://eps.leeds.ac.uk/computing/stafflist>). The information for each supervisor was selectively extracted from their personal web pages in ‘People’ page, includes:

Areas of Expertise: This section, which provides a brief overview of the supervisor's field of expertise, was considered key as it would determine their respective areas of specialization, especially for those who had not provided any other information.

Profile: This section was more varied as the supervisors did not follow a template to write this section. The collected data may include personal introductions, research funding, current and past projects, students and their projects, student feedback and potential postgraduate research opportunities.

While the 'Publications' section was initially considered for scraping, it was ultimately excluded. The content of this section, which mostly comprised of paper titles, publishers or conference names, and authors, was found to potentially introduce noise and sparsity into the dataset in following experiments. This could interfere with the vectorization and recommendation processes. However, it was observed that the topics of the publications generally aligned with the supervisors' areas of expertise, mitigating the impact of the exclusion.

The web scraping process finally built a dataframe consisting of information from 101 supervisors.

## 4.2 Data Cleaning

The raw data obtained from the scraping process required several cleaning steps:

Removal of HTML tags, special characters, and long white spaces: The raw scraped data contained various HTML tags, special characters, and excess white spaces, which were removed to obtain clean text data.

Expanding abbreviations: the dataset contains many abbreviations and these need to be expanded, e.g. ‘I'm’ should be replaced with ‘I am’, which ensures that ‘I'm’ and ‘I am’ are not judged to be two things during word vectorisation.

Removal of common terms: Certain common terms, such as 'publications', 'Research projects', 'Profile', and 'Research groups and institutes', were removed. These words or phrases were found to interfere with topic modelling. For example, when they are not removed, most of supervisors’ topics will contain the word ‘Profile’, and then the topics with those words will replace the original position of the computer field topics.

Removal of System-Generated sentences: Certain phrases, such as "I am currently working on will be listed below.", were automatically generated by the system on the pages and included in all supervisor profiles, regardless of whether the supervisor had added any information. By removing these phrases, supervisors who had not written any information could be identified and excluded from the dataset.

After applying these cleaning steps, the cleaned dataframe contained information from 80 supervisors.

In addition to these cleaning steps, an alternative cleaning strategy was applied to create a second version of the dataset, where all pronouns and conjunctions (such as 'I' and 'and') were removed. This variant will be used in subsequent experiments to investigate the effects of removing these words when vectorizing the text data using CountVectorizer, due to this technique focuses on word frequency when vectorise words.

## 4.3 Recommendation Algorithms Analysis

In this stage, the focus was on comparing the performance of two recommendation algorithms: cosine similarity and Jaccard similarity.

## 4.3.1 Jaccard Similarity Analysis

The Jaccard similarity algorithm views each text (both supervisor information and user input) as a set of words. It then calculates the Jaccard similarity between these sets to measure the similarity between the texts. To apply the Jaccard similarity algorithm, all words of supervisors’ information and user input were converted into a set of words. The Jaccard similarity scores between supervisor’s sets and user input’s set were computed to give each supervisor a recommendation score based on a certain input. The supervisors were then ranked according to their scores, with higher scores indicating a higher recommendation.

## 4.3.2 Cosine Similarity Analysis

For the cosine similarity algorithm, the first step was to vectorize the words using CountVectorizer. The details of settings in CountVectorizer will be discussed in the following section. The user input was also vectorized in the same way. The cosine similarity between the user input vector and each supervisor's vector was then calculated, providing each supervisor with a recommendation score.

And then supervisors were also then ranked according to their scores. All of the next experiments will only consider whether the top 5 recommended supervisors satisfy the user inputs.

## 4.3.3 Comparison and Results

After testing both algorithms with the same user input, it was found that the cosine similarity algorithm provides more accurate recommendations than the Jaccard similarity algorithm. The results of these experiments are demonstrated and discussed next.

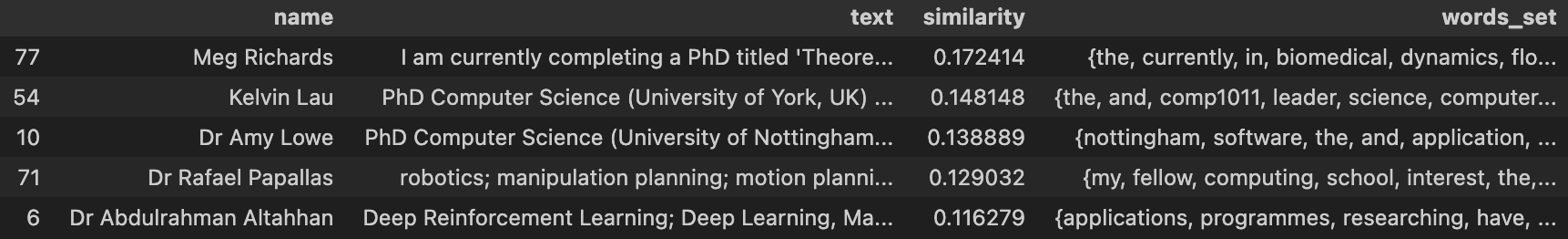
## 4.3.3.1 Experiment 1

- User Input: "I am looking for an expert in data mining and text analysis.",

- Cosine Similarity Results: 4 of the top 5 supervisors recommended mention 'data mining' or 'text analysis' on their profile.

- Jaccard Similarity Results: Only found a supervisor of the top 5 which ranked fifth, Dr. Abdulrahman Altahhan, with a ‘data analysis’ background.

- Discussion: Jaccard similarity algorithm recommend Dr. Abdulrahman Altahhan because the wordset misinterpret ‘data mining and text analysis’ as ‘data analysis’. And Professor Roy Ruddle was recommended by cosine similarity algorithm because he mentioned ‘Data Analytics’ on his profile. The vectorization method made a similar mistake.



**Figure 4.3.3.1.1 Jaccard similarity results**

Input "I am looking for an expert in data mining and text analysis."



**Figure 4.3.3.1.2 Cosine similarity results**

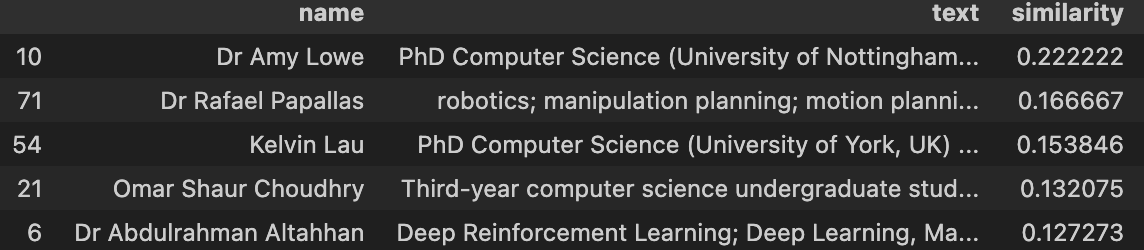
Input "I am looking for an expert in data mining and text analysis."

## 4.3.3.1 Experiment 2

- User Input: "I have 5 years of software development experience and I am looking for a supervisor who is familiar with web development and software engineering to help me with my final project."

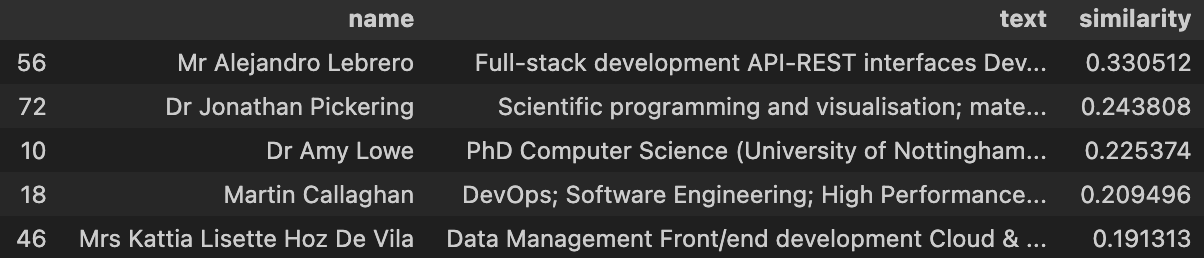
- Cosine Similarity Results: All the 5 supervisors have the background fit the requirements.

- Jaccard Similarity Results: Only 3 supervisors who are eligible (Dr Abdulrahman Altahhan and Omar Shaur Choudhry do not meet the requirement).



**Figure 4.3.3.3 Jaccard similarity results**

Input "I have 5 years of software development experience and I am looking for a supervisor who is familiar with web development and software engineering to help me with my final project."



**Figure 4.3.3.4 Cosine similarity results**

Input "I have 5 years of software development experience and I am looking for a supervisor who is familiar with web development and software engineering to help me with my final project."

After conducting lots of similar tests, the cosine similarity algorithm always shows more accurate results than the Jaccard similarity algorithm. Cosine similarity algorithm recommends the top 5 supervisors with an average accuracy of greater than 80%, while the Jaccard similarity algorithm's average accuracy is less than 60%.

These results demonstrated the superior performance of the cosine similarity algorithm for this specific recommendation task. It also demonstrates that the recommendation approach using word vectorization performs better on this task than the recommendation approach using word set, so the following different vectorization techniques would be tested.

## 4.4 Vectorization Techniques Analysis

This phase of the implementation involved testing several methods for text vectorization. All of those vectorization methods were used in cosine similarity recommendation approach.

## 4.4.1 Bag-of-Words Models

## 4.4.1.1 CountVectorizer

The first method tested was CountVectorizer. It vectorizes text by counting the frequency of each word in the text.

The ngram\_range parameter in CountVectorizer method was set to (1, 2) to consider both single words and two-word phrases as separate features in the vectorization process. This is because there are so many two-word phrases in the text, such as 'machine learning' and 'data mining', which are common phrases in computing field, and vectorizing them separately can lead to deviation from the original meaning. Setting parameter to (1, 3) or a larger range has also been tested, and they differ from (1, 2), only slightly in the ranking of results, and (1, 2) has optimal results in most case. So, in next experiments, the parameter of ngram\_range was all set to (1, 2).

After testing, the system using this method typically assigns high scores to certain supervisors such as Sharib Ali and Noorhan Abbas, who have detailed and longer text content. This is because CountVectorizer vectorize words by their frequency, and supervisors with long content naturally have a higher frequency of certain keywords.

While this bias towards supervisors with more content might be acceptable in some cases, it can also lead to less accurate recommendations when the user input is more general. Therefore, it may overemphasize words that frequently appear in all contents, such as common pronouns, and conjunctions. So, another experiment was set to test this hypothesis.

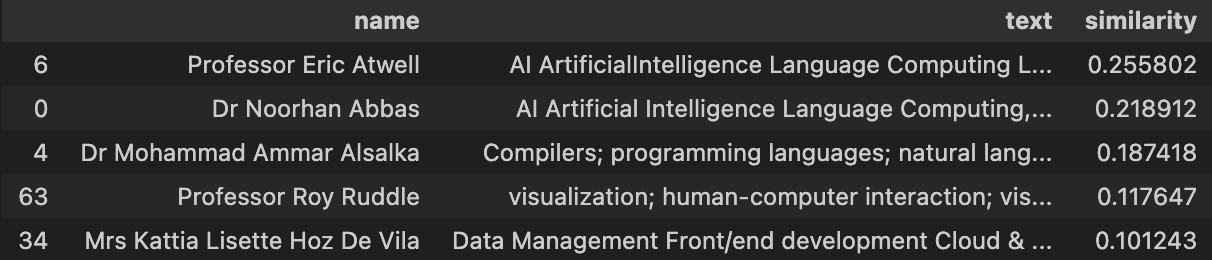
In another experiment, these common words were removed from the dataset to minimize their impact on the recommendation system. The tested results from using CountVectorizer were comparable to those obtained with the cleaned dataset where all pronouns and conjunctions were removed. The resulting list of supervisors differed only slightly. In the following example:

- User Input: "I am looking for an expert in data mining and text analysis.",

- Original Dataset: 4 of the top 5 supervisors recommended mention 'data mining' or 'text analysis' on their profile.

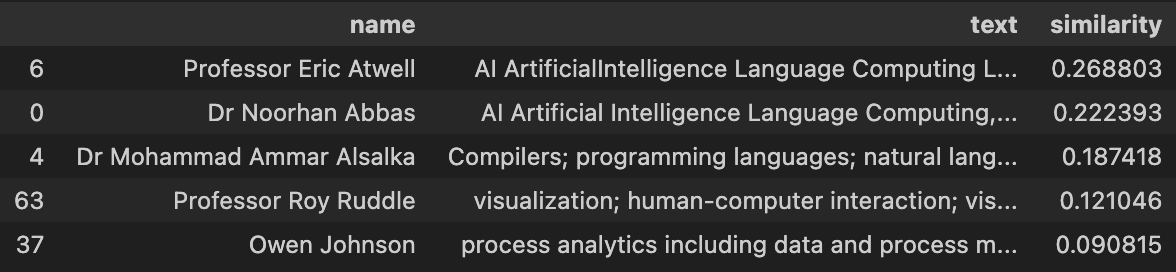
- Cleaned Dataset: 4 of the top 5 supervisors recommended mention 'data mining' or 'text analysis' on their profile.

- Discussion: Original dataset and cleaned dataset gave same results in same order in first 4 supervisors, only the last supervisor was different, Mrs. Kattia Lisette Hoz De Vila and Owen Johnson respectively. Neither of them can be counted to have a strong background in data mining and text analysis. So, the experiment proved that pronouns, and conjunctions do not affect the results of the CountVectorizer.



**Figure 4.4.1.1.1 Original dataset results**

Input "I am looking for an expert in data mining and text analysis."



**Figure 4.4.1.1.2 Cleaned dataset results**

Input "I am looking for an expert in data mining and text analysis."

Hence, this project would continue with the original cleaned dataset.

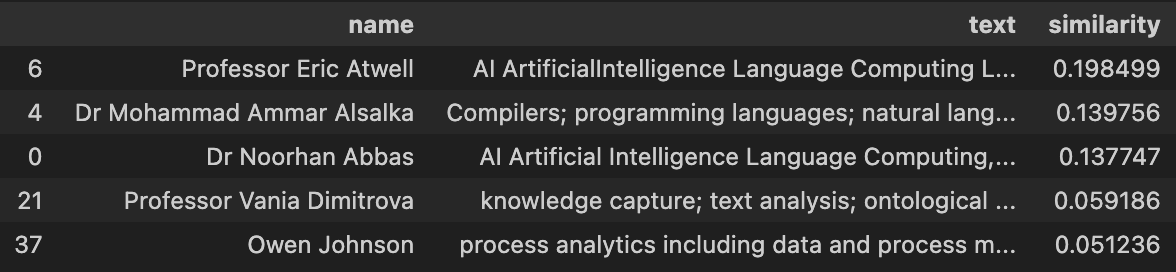
## 4.4.1.2 TF-IDF

The second method tested was Term Frequency-Inverse Document Frequency (TF-IDF), which provides a measure of relevance of words in the document by penalizing common words across all contents and highlighting words that are unique to a particular document.

- User Input: "I am looking for an expert in data mining and text analysis.",

- TF-IDF: All 5 supervisors mention “data mining” or “text analysis” in their profile and the rank of these 5 supervisors is reasonable.

- Discussion: The top 5 supervisors recommended by TF-IDF were similar to those recommended by CountVectorizer with cleaned dataset, with only minor differences in the rankings (Dr Mohammad Ammar Alsalka and Dr Noorhan Abbas switches position, Professor Vania Dimitrova is recommended but Professor Roy Ruddle is not). This suggests that TF-IDF achieve the same effect as cleaning the dataset manually by penalizing common words. And that Professor Vania Dimitrova fits the context described by the user better than Professor Roy Ruddle. So, TF-IDF performs slightly better than CountVectorizer on this task.



**Figure 4.4.1.1.2 TF-IDF results**

Input "I am looking for an expert in data mining and text analysis."

## 4.4.2 Word Embedding Models

## 4.4.2.1 Word2Vec

The next technique tested was Word2Vec, a word embedding method. Two different models were used: a self-trained model using the supervisor texts and a pre-trained model (word2vec-google-news-300) from Google.

After tested, the self-trained model does not perform as well as the pre-training models. However, the pre-training models have a weakness that it may not be vectorized some words due to the texts contains words that are not trained in these models and have to drop them, which ultimately leads to poor results.

Therefore, after tested, only the word2vec-google-news-300 model can vectorize all the words in the dataset and it also performs better than the self-trained one. So, it is used as the final model for Word2Vec.

Moreover, Word2Vec recommendation approach tend to rank Amy Lowe at first place for inputs containing 'software engineering and web application', because she directly mentioned this phrase in her profile. However, her profile only has one sentence.

## 4.4.2.2 Doc2Vec

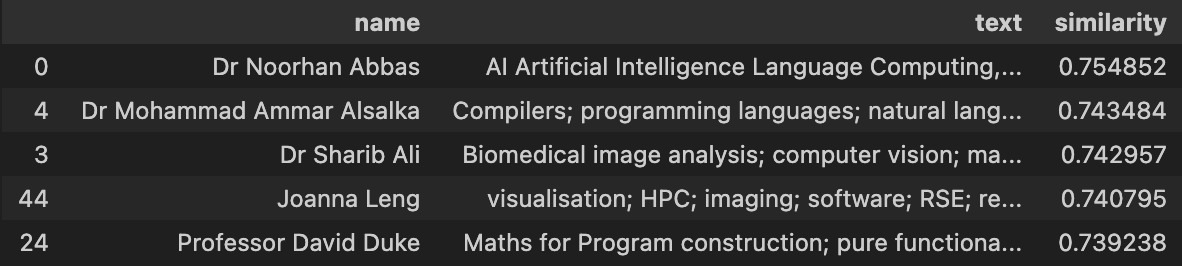
The last technique tested was Doc2Vec, an extension of Word2Vec that considers the document as a whole. In same input, the Doc2Vec model did not rank Amy Lowe as high as the Word2Vec model did for the same input. Instead, it recommended supervisors with longer profiles and relevant backgrounds. This suggests that Doc2Vec may have a better understanding of the overall semantic context, especially for longer texts.

- User Input :"I am looking for an expert in data mining and text analysis.",

- Word2Vec Results: Only the first 2 (Dr Noorhan Abbas and Dr Mohammad Ammar Alsalka) of the 5 supervisors recommended by Word2Vec met the requirements.

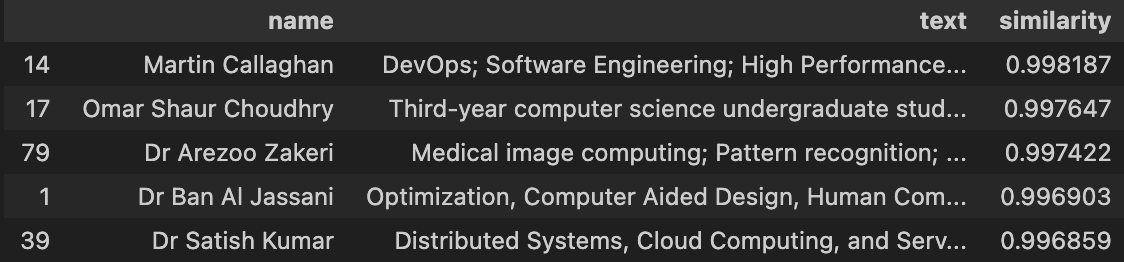
- Doc2Vec Results: Only the first (Martin Callaghan) of the 5 supervisors recommended by Doc2Vec met the requirements.

- Discussion: This suggests that despite most of the supervisor’s information is long text, Doc2Vec recommendation does not improve compared to Word2Vec, but instead regress.



**Figure 4.4.2.2.1 Word2Vec results**

Input "I am looking for an expert in data mining and text analysis."



**Figure 4.4.2.2.2 Doc2Vec results**

Input "I am looking for an expert in data mining and text analysis."

## 4.4.3 Experiments and Results

After multiple testing, strengths and weaknesses of the four vectorization techniques are found. The bag-of-words models (CountVectorizer and TF-IDF) provided consistent and reliable results, making them suitable for this task. The word embedding models (Word2Vec and Doc2Vec), while capable of capturing semantic relationships between words, showed some limitations, particularly when dealing with domain-specific terms.

To further verify the effectiveness of these vectorization techniques, an additional experiment was conducted. A summary of Dr. Ban Al Jassani's information was generated using the Bart and used as user input. The goal was to see if the recommendation system, using each of the four vectorization methods, could accurately recommend Dr. Ban Al Jassani as the first place.

The results showed that CountVectorizer and TF-IDF successfully ranked Dr. Ban Al Jassani as the top recommendation. However, the self-trained Word2Vec and Doc2Vec models failed to rank him highly, and the Word2Vec model using word2vec-google-news-300 model only ranked him third. These results further confirmed the superior performance of the bag-of-words models for this particular recommendation task.

## 4.5 Text Generation Tools Analysis

In this section, three different text generation techniques, TextRank, BART and T5 are deployed to summarize the information of various supervisors.

**4.5.1 TextRank**

TextRank does not directly control the generated summary length. It merely constructs a graph based on sentence similarities and ranks sentences using the PageRank algorithm. Consequently, the summaries generated by TextRank were too brief, consisting of a single sentence, making this method unsuitable for our needs.

**4.5.2 BART and T5**

The transformer-based models, BART and T5 delivered impressive performance. By setting the parameter ‘min\_length’ to 10 and ‘max\_length’ to 100, these models were able to generate readable summaries of moderate length. However, both of these models rely on pre-trained models, In this experiment, T5-large and BART-large was chosen to use. And as described in their respective papers and documentation, architecture for T5 and BART is as follows:

**Table 4.5.2.1** Architecture of T5 and BART

|  |  |  |
| --- | --- | --- |
| **BART-large** | **T5-base** | **T5-large** |
| 12-encoder layers  12-decoder layers  1024-hidden size | 12-encoder layers  12-decoder layers  768-hidden size  220M parameters (approximately 2 \* bert-base) | 24-encoder layers  24-decoder layers  1024-hidden size  770M parameters |

The T5-large model is approximately twice as large as the BART-large model. Correspondingly, after tested, the training time for T5-large was more than twice that of BART-large. Despite using the 'large' models and the same `max\_length` and `min\_length` parameters for both, the summaries generated by T5 were consistently longer than those produced by BART. However, the quality of content generated by T5 was found not as good as BART. For instance, in the case of Dr. Mark Walkley's summary, which includes the name of Evangelia Antonopoulou, T5 incorrectly made Evangelia Antonopoulou as the subject in the summary. On the other hand, BART did not make this mistake, demonstrating its superior performance in this context.

**Table 4.5.2.2** Example of T5 and BART generated summary

|  |  |
| --- | --- |
| Bart generated | T5 generated |
| “Mark is a member of the Computational Science and Engineering research theme. His areas of teaching range from introductory programming to computer networks and parallel computing.” | “Evangelia Antonopoulou is a member of the Computational Science and Engineering research theme. He has taught at every year of study on our taught programmes. His areas of teaching range from introductory programming to computer networks and parallel computing.” |

In summary, while both BART and T5 demonstrated their capabilities as text summarization tools, BART is more reliable and efficient for this specific task.

## 4.6 Topic Modelling Techniques Analysis

This section investigates three different topic modelling techniques: Latent Dirichlet Allocation (LDA), Non-negative Matrix Factorization (NMF), and BERTopic. These unsupervised learning methods aim to automatically uncover the hidden thematic structure in the documents.

## 4.6.1 LDA and NMF

The method begins with vectorization using CountVectorizer, followed by a further text cleaning step to delete terms such as 'university', 'leeds', 'uk', 'phd', 'postgraduate'. Otherwise, these terms might be assigned as topics for supervisors. Setting LDA algorithm to generate 20 topics, because too small number of topics would recommend a big cluster to users which includes some supervisors do not meet user’s requirement. User input takes the same LDA process, and supervisors that best match the user's topics are recommended. This method only recommended a limited number of supervisors, the results are similar to the cosine similarity method. However, if the user input did not align with the allocated topics, this algorithm cannot give any useful recommendation.

And then, using the same method as with LDA, the NMF algorithm was tested. However, in most cases, LDA recommended more useful results than NMF, for example, when input "I am looking for an expert in data mining and text analysis.", LDA recommended six supervisors, while NMF recommended only two. The reason for this may be that LDA is a probabilistic model that generates probabilistic explanations for the data. This may make LDA more able to handle uncertain and noisy dataset which just like the dataset of this experiment.

## 4.6.2 BERTopic

At last, this experiment tested the BERTopic. The process of building BERTopic is similar to LDA, only need to set the ‘min\_topic\_size’ parameter. This parameter controls the minimum number of supervisors per topic cluster. However, the default setting is 10, resulting in only 3 clusters for all supervisors (1, 0, and -1, -1 means an ineffective noise category). This was inadequate for the recommendation task. When this parameter was set to 2, 12 categories were obtained, but the final recommendation results were not satisfactory. Because lowering ‘min\_topic\_size’ can lead to overfitting, as some topics containing a small number of supervisors may be noise or accidental clusters, rather than real topics.

While there are some pre-trained BERTopic models on Hugging Face (Hugging Face, 2023). This includes classification of IMDB movies, classification of academic paper content, but they are all trained on thousands of articles and documents. However, this project cannot provide such a large dataset for training. So, BERTopic model is not suitable for this task.

## 4.6.3 Conclusion

In summary, while these topic modelling techniques have their merits, their limitations and effectiveness can vary depending on the data characteristics and specific task. In this case, LDA provided the most reliable results for the supervisor recommendation task.

## 4.7 OpenAI Techniques Integration

This section explores the integration of OpenAI techniques into the supervisor recommendation system, drawing inspiration from the topic modelling approaches. The idea is to use the text generation API of OpenAI to extract topics from the supervisors' information and then match these topics with user's input using vectorization and cosine similarity approach.

## 4.7.1 text-davinci-003

OpenAI provides a paid API, text-davinci-003, which is a language model used to generate human-like texts just like ChatGPT. This model requires a prompt to guide the generation process. The following prompt was used:

"I am sending you an introductory information of a computer science tutor. Please help me to find some topics in this information about the field of computing that can be summarized in just a few words, e.g., machine learning, deep learning. Please separate all topics with commas, like 'machine learning', 'deep learning'. And return 0 if no suitable topic is found. It is best to keep tutor topics to between 1 and 6. If the next topic you want to generate exceeds the token limit, stop the generation of this topic. ?\n Information: {text}\n Topic:"

The generated topics and user input were vectorized using the TF-IDF method. The results showed significant improvements. Also using this input: "I am looking for an expert in data mining and text analysis." The result is in first 8 recommended supervisors, 7 meet this input’s requirement.

## 4.7.2 text-embedding-ada-002

OpenAI also provides a powerful embedding API, text-embedding-ada-002. This API was used to vectorize the supervisors’ information and the user’s input, and cosine similarity was then calculated. The results were outstanding.

After attempting a more complex user input, "I have 5 years of software development experience and I am looking for a supervisor who is familiar with web development and software engineering to help me with my final project. ", the cosine similarity algorithm gives a list of the eight supervisors which all of them expertise in software/web development.

The results were also comparable to those obtained by summarizing the text with OpenAI and then vectorizing it using TF-IDF. These two methods emerged as the best for the supervisor recommendation system.

However, the use of the embedding API does have a disadvantage. In addition to the cost for vectorizing all supervisors’ information, every user input also needs to be vectorized via the API, leading to an additional charge and this process making the system has to access the internet to do recommendation. While the cost is minimal, it might not be practical to make a request to OpenAI and incur a charge each time a user uses the system.

User input vectors generated using previous vectorization methods were also used to test compatibility with vectors generated by this API, thus avoiding charges and network connections of user input vectorization every time. Unfortunately, none of the previous vectorization methods are compatible with the vectors produced by this API.

As a result, this method is difficult to deploy on a real system compared to the method of generating topics using text-davinci-003 which only need to pay once and doesn’t have charges and networking requirements for everyday use.

## 4.8 Clustering Recommendation Analysis

This section explores the implementation of a clustering-based recommendation system. After standardization and PCA dimensionality reduction on embeddings, KMeans clustering is applied and selected clustering number with silhouette analysis. This clustering is then used to recommend supervisors from the same cluster, effectively tailoring recommendations to individual users.

## 4.8.1 Data Pre-processing

The first stage of the process involves pre-processing the data. The embeddings which obtained from text-embedding-ada-002 are initially standardized using the `StandardScaler` function, which adjusts the values so that each feature has a mean of 0 and a standard deviation of 1. This standardization process is important for eliminating scale differences between features.

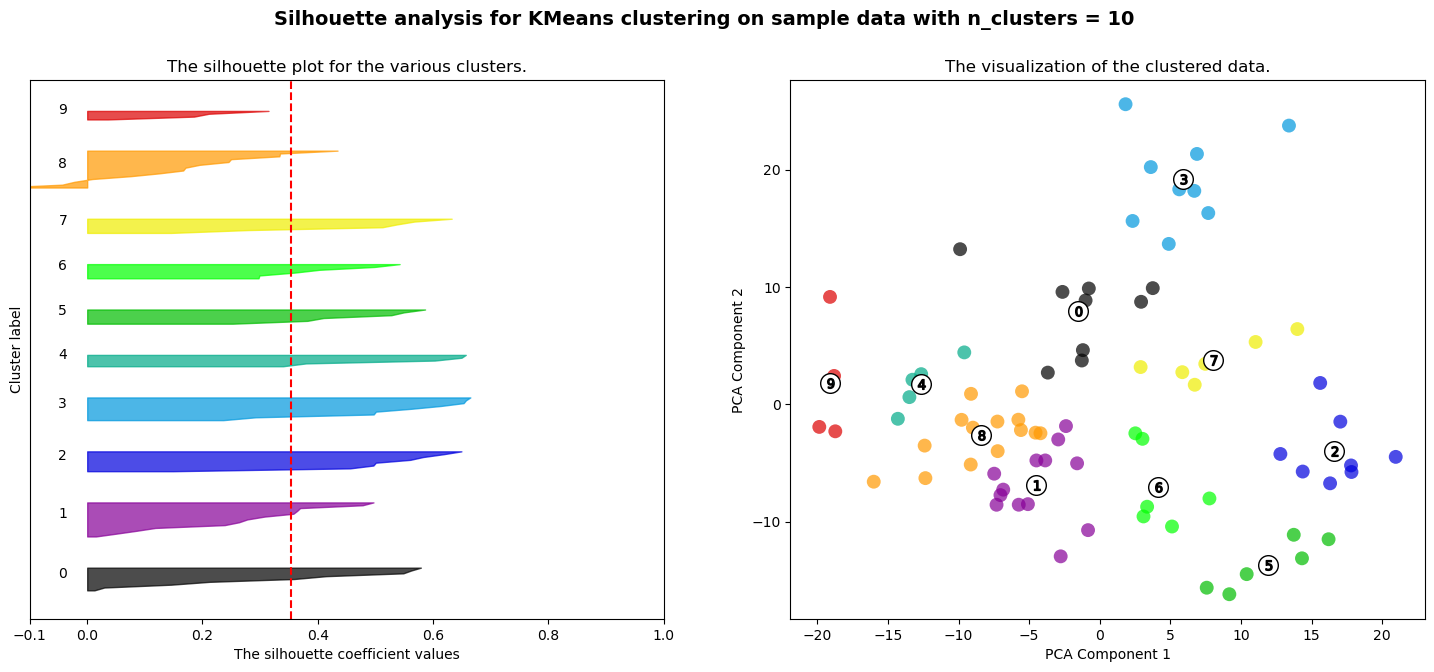
Following standardization, a Principal Component Analysis (PCA) is performed on the embeddings. PCA is a popular technique for data dimensionality reduction that maps multi-dimensional data onto a lower-dimensional space, while striving to retain as much of the original data structure and distribution information as possible. In this case, the embeddings are initially reduced to a two-dimensional space using PCA, before being clustered. Turning embeddings into two dimensions also makes it easier to do visualization.

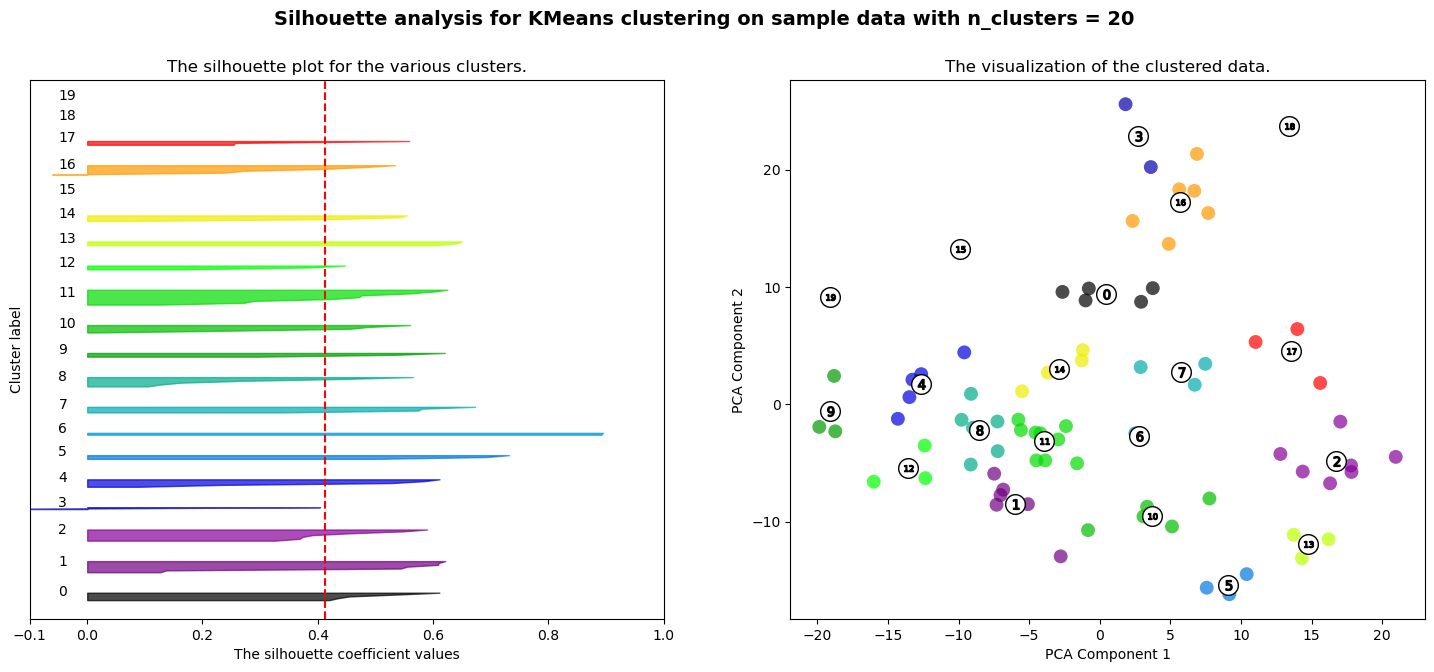
## 4.8.2 Clustering Analysis

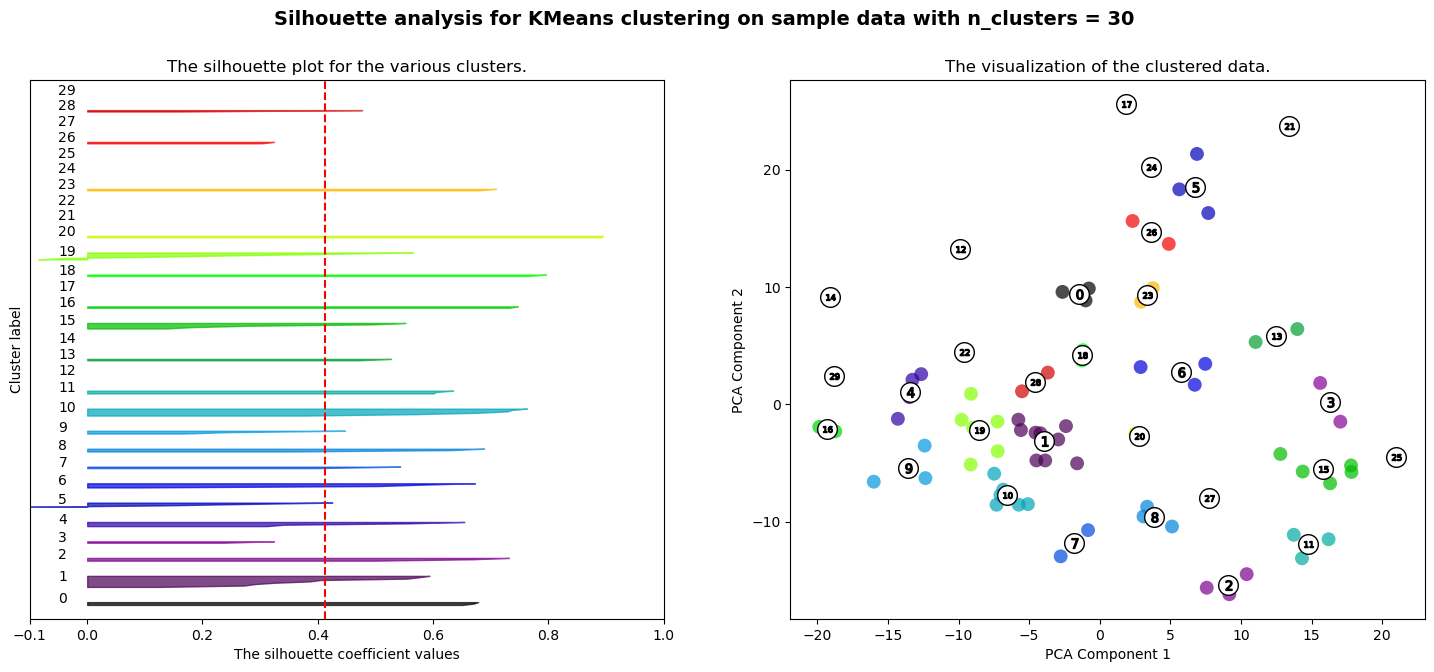
The next stage involves conducting a clustering analysis (scikit-learn developers, 2023). Different numbers of clusters are tested, and the silhouette coefficient is used to evaluate the effectiveness of each cluster number. The silhouette coefficient is a measure of cluster effectiveness, with values ranging from -1 to 1. Higher values indicate more effective clustering, in this case, meaning that each supervisor's background is more similar within a cluster. The silhouette coefficient is calculated for each cluster number. It is found that when the number of clusters is 20, the silhouette score (coefficient) is the highest, indicating that 20 is the most suitable clusters number for supervisors.



**Figure 4.8.2.1 Silhouette scores**







**Figures 4.8.2.2 Silhouette analysing images**

The silhouette plot on the left shows which silhouette coefficient values are highest for each number of clusters, and the red line is their average value, which is highest when the clusters number is 20. The scatter plot on the right shows each cluster.

## 4.8.3 Supervisor Recommendation

After determining the optimal number of clusters, KMeans clustering is used to cluster the data. Subsequently, supervisors who are in the same cluster will be recommended to the user when they choose a supervisor. For instance, if a user selects 'Professor Eric Atwell', the system will return a list that includes 'Dr Noorhan Abbas' and 'Dr Abdulrahman Altahhan', as these three individuals all have similar academic backgrounds or areas of expertise.

# Chapter 5 Evaluation

This section provides an evaluation of the different approaches explored in the implementation of the supervisor recommendation system.

## 5.1 Recommendation Approaches

Cosine Similarity: The cosine similarity approach performed reasonably well in generating user-relevant recommendations. It was effective in capturing the semantic similarity between the users' requirements and the supervisors' profiles. However, it was limited by its reliance on direct term matching, which may not always capture the nuanced similarities in academic interests.

Jaccard Similarity: The Jaccard similarity approach, driven by the notion of shared words between documents, was not chosen due to its limited applicability compared to cosine similarity. Most of the subsequent techniques were driven by word vectors, and cosine similarity, which operates on vectorized representations, showed better compatibility with them. Moreover, cosine similarity also demonstrated superior performance in recommendation accuracy compared to Jaccard similarity.

## 5.2 Vectorization Methods

The vectorization method plays a crucial role in the performance of the recommendation system. Among the different methods tested, the Bag of Words models, particularly CountVectorizer and TF-IDF, show the best performance.

## 5.2.1 Bag of Words models

CountVectorizer: Despite its excellent performance, the CountVectorizer showed some limitations. For instance, it struggled in discerning between the use of the same word in different contexts, such as "visual" in Visual Basic and Visual C++. This often led to irrelevant recommendations, such as supervisors with expertise in visual analytics, when the user's interest is in programming languages.

TF-IDF: The TF-IDF vectorizer performed exceptionally well due to its ability to weight words based not only on their frequency in a document but also on their overall frequency across all documents. This allowed the model to better differentiate between common and distinctive words, resulting in more accurate recommendations.

## 5.2.2 Embedding models

Word2Vec and Doc2Vec: Although these models, which require pre-trained embeddings, provided acceptable results, they fell short in understanding domain-specific terms due to their training on general corpora. For instance, Word2Vec's interpretation of the term 'text analysis' might be too broad, leading to recommendations of supervisors with expertise in other type of analysis, not only 'text analysis’. It's like the incorrect recommendation problem that CountVectorizer gives, but they have different principles.

OpenAI's text-embedding-ada-002: This model stands out among the embedding models for the task. It was able to capture complex semantic relationships due to its training on a very large corpus. However, its application was limited by the need for internet access and additional charges for each vectorization.

## 5.2.3 Summary

In conclusion, while each of the vectorization methods had its strengths, the choice of method largely depended on the specific task requirements and the nature of the data. For this specific application, the OpenAI's text-embedding-ada-002 and TF-IDF were identified as the most suitable due to their superior performance in capturing semantic similarities and their compatibility with the subsequent techniques used in the recommendation system.

## 5.3 Topic Modelling Techniques

Among the tested topic modelling techniques, Latent Dirichlet Allocation (LDA) provided the most reliable results. Non-negative Matrix Factorization (NMF) was less effective in recommending a diverse set of relevant supervisors, while BERTopic was not suitable due to the limitations of the data set size and overfitting issues.

Based on the ideas of topic generation, a OpenAI technique, text-davinci-003 API was tested. It used to generate topics from the supervisor’s information, showed significant improvements when combined with TF-IDF vectorization.

## 5.4 Clustering Recommendation Analysis

The clustering-based approach using text-embedding-ada-002 embeddings, standardization, PCA, and KMeans clustering turned out to be an effective method. The approach was able to group supervisors into meaningful clusters, which then facilitated efficient and relevant supervisor recommendations.

## 5.5 Summary

Overall, in terms of recommendation approaches, the cosine similarity method, deployed with TF-IDF or text-embedding-ada-002 vectorization, emerged as the most effective. The choice between TF-IDF and text-embedding-ada-002 would depend on budget consideration and server capability. Alternatively, using topics generation by the text-davinci-003 API, followed by TF-IDF vectorization and cosine similarity, also produced excellent results and no need to pay OpenAI for daily uses. For text summarization, the BART tool proved its effectiveness in generating concise and informative summaries for each supervisor. Lastly, a clustering recommendation approach based on KMeans was implemented, enhancing the practicality of the system by recommending similar supervisors from the same cluster.

Therefore, after a thorough evaluation, final project should be deployed with the following technologies: TF-IDF or text-embedding-ada-002 vectorization, cosine similarity algorithm, BART text generation and KMeans clustering recommendation.

## 5.6 Hypothetical Use Cases

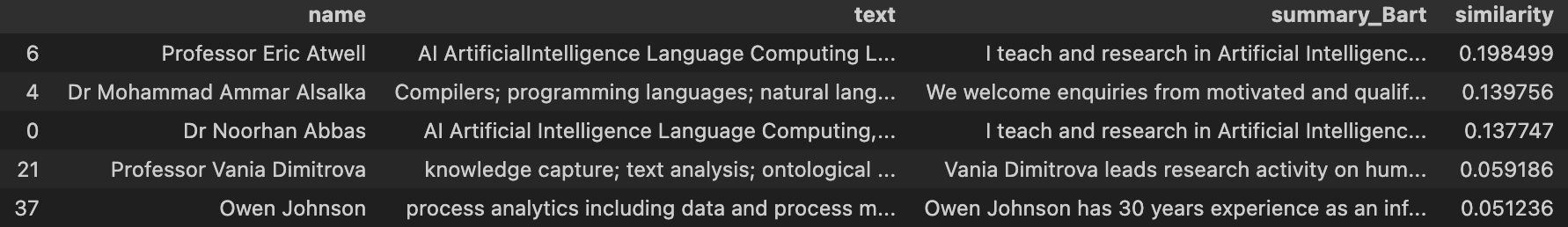
Due to the lack of real users to test the system, hypothetical user evaluation process is designed. Each user is given a unique scenario to find their suitable supervisor. The system will use TF-IDF as vectorization method.

Evaluation Metrics: After users input their interests or project descriptions, then the relevance of top 5 recommended supervisors based on their expertise and interests will be evaluated.

## 5.6.1 Use case 1

A student Alice finds supervisors for her final project, she inputs: "I am looking for an expert in data mining and text analysis."

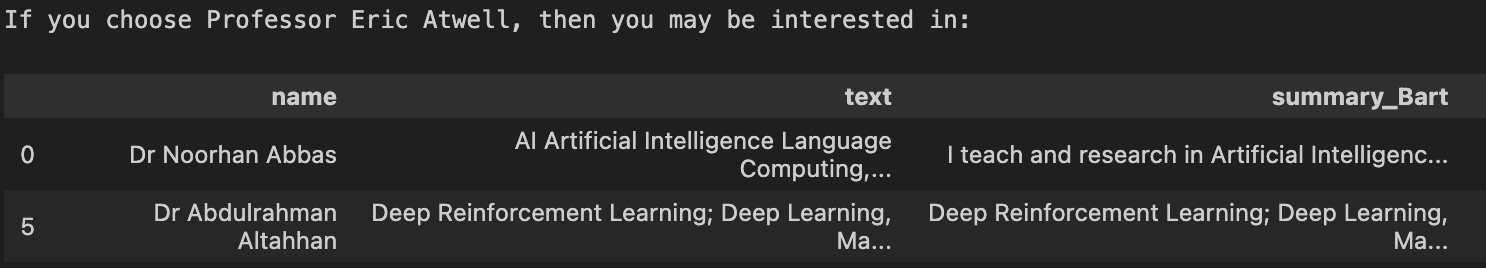
Results:



**Figures 5.6.1.1 Recommendation result for Alice**

All 5 supervisors are expertise in data mining and text analysis, after read their introduction (summary\_Bart), Alice choose Professor Eric Atwell as he is ranked No.1.

The system then suggests similar supervisors for Alice from Professor Eric Atwell:



**Figures 5.6.1.2 Clustering recommendation for Alice**

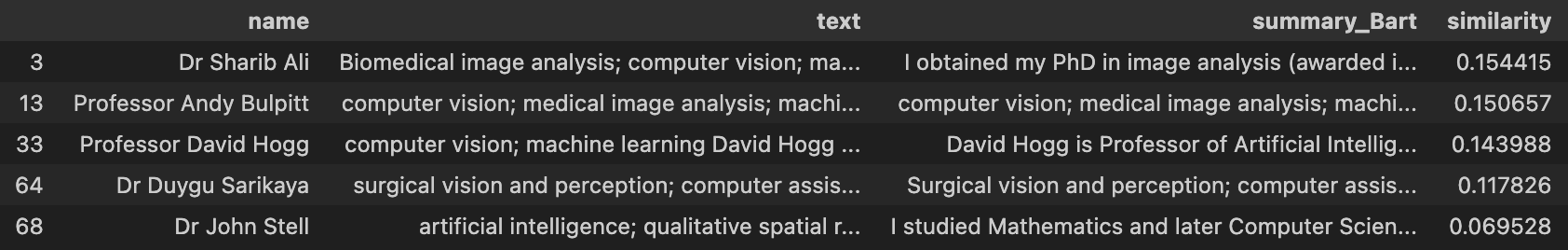
After Alice choose Professor Eric Atwell, the system also recommends Dr Noorhan Abbas and Dr Abdulrahman Altahhan to Alice. So, Alice chose them as well. Eventually, Alice chose three supervisors.

In this case, the similarity scores of the 5 supervisors recommended by cosine similarity are ranked into 3 bands: 0.19, 0.13 and 0.05. Professor Eric Atwell in the first band is undoubtedly the supervisor that best matches user's requirements. The two supervisors in the second band, Dr Mohammad Ammar Alsalka and Dr. Noorhan Abbas, are both mainly interested in NLP technologies, which also meets the user's requirements. In the third band, Professor Vania Dimitrova is also a supervisor who expertise in text analysis. The fifth tutor, Owen Johnson, only mentions data mining in his Areas of Expertise and doesn't mention anything else relevant in his profile. So, the results of this recommendation system are very reasonable.

## 5.6.2 Use case 2

A faculty member, Bob, is looking for a collaboration with colleagues who have expertise in computer vision and C++. He inputs: "I am currently working on a study about computer vision. I need some experts in this field to collaborate with me. Also, anyone who is proficient in C++ can join me in this project."

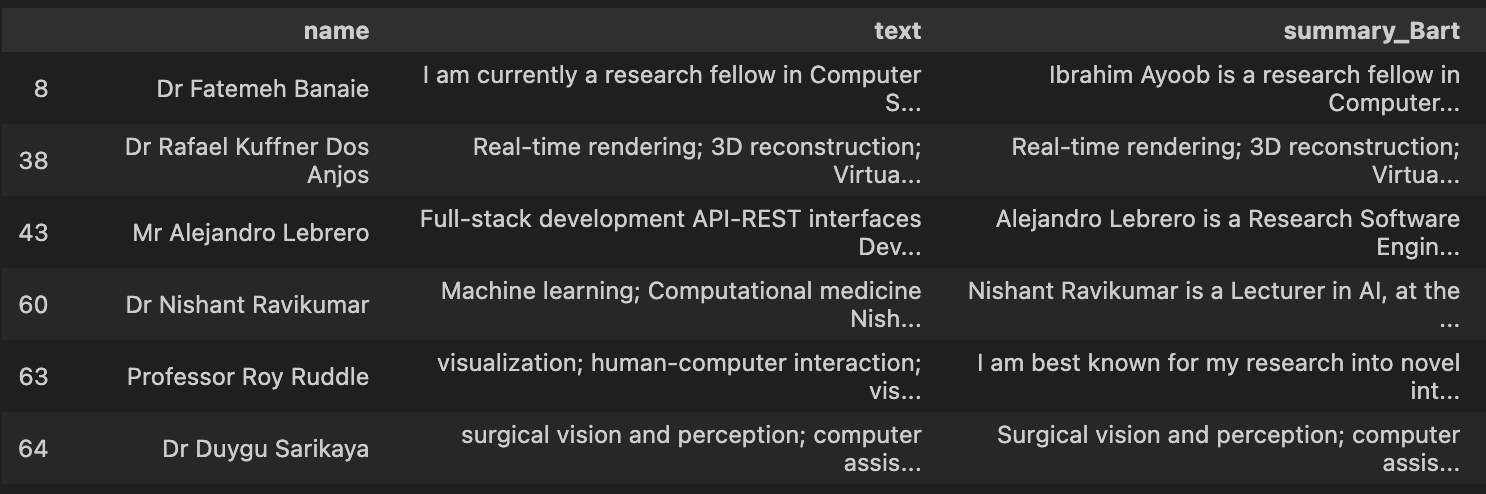
Results:



**Figures 5.6.2.1 Recommendation result for Bob**

4 of 5 colleagues are expertise in computer vision and he has 4 options now, but Bob cannot find a colleague mentioned about C++.

So, he tries to find colleagues of these people who are proficient in C++. And then he chooses Dr Sharib Ali, as he is ranked No.1, then the system suggests him as following:



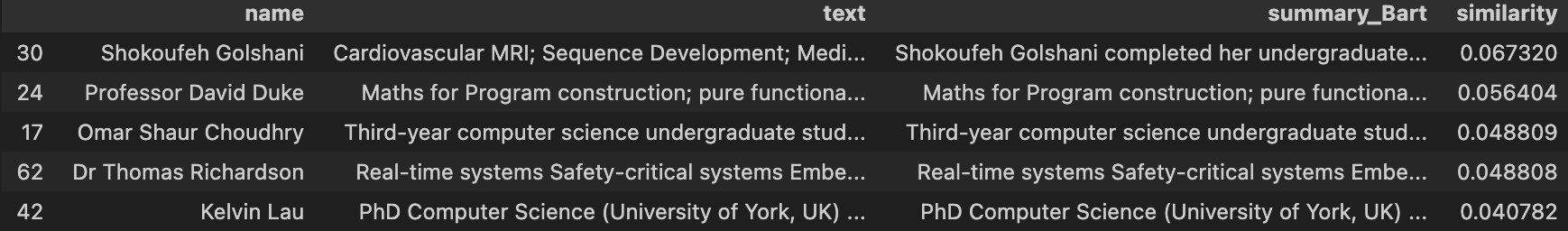
**Figures 5.6.2.2 Clustering recommendation for Bob**

After he checked all those 6 colleagues’ introduction, he chooses Dr Fatemeh Banaie as she mentioned she taught C++ course before. So, Bob totally chooses 2 colleagues and has many alternative options.

In this case, the similarity scores also show the relevance of the supervisors’ background to the user’s requirements. The first 4 supervisors are in the 0.11-0.15 score range, all of them meet the user's requirements for a computer vision background. Only the last supervisor, Dr John Stell, does not mention anything in computer vision field, and his score is 0.06 which obviously lower than the previous supervsiors.

## 5.6.3 Use case 3

A student does not have a thought for his final project, so he input this "I love programming, but I don't have any ideas for my project at the moment and I would like to find a supervisor who can help me."

Results:

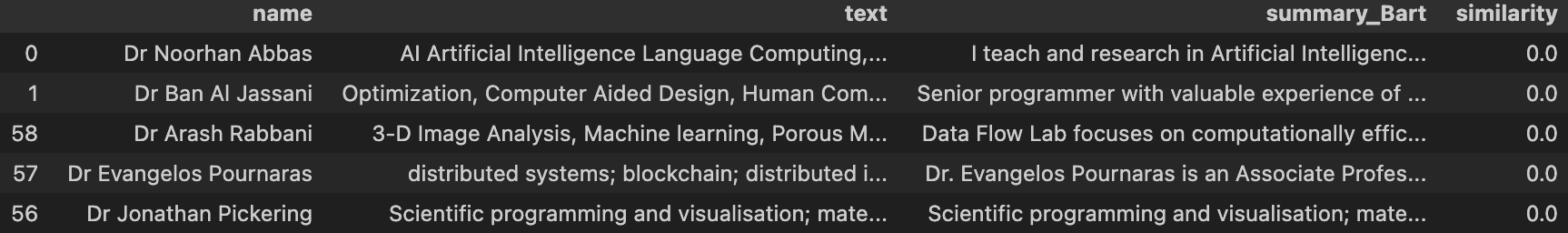
**Figures 5.6.3.1 Recommendation result**

The system still recommends him supervisors. But their similarity scores are much lower compared to the recommendations with clear requirements in the first two cases.

Except the first supervisor, Shokoufeh Golshani, other 4 supervisors all mentioned that the areas of their research include the field of programming or they have taught programming courses. So given that this is a text-analysis-based recommendation system, the system is always able to find a certain keyword in the input and matching suitable targets.

However, when user inputs do not contain a keyword that has appeared in the dataset, the system will not make any recommendation for user, and the similarity score for all supervisors will be 0.

Results:



**Figures 5.6.3.2 Recommendation result**

When user inputs “I love music.” or “I like dogs.”

# Chapter 6 Conclusion & Future Work

## 6.1 Conclusion

This project aimed to investigate and evaluate various machine learning and natural language processing techniques for developing a reliable supervisor recommendation system for students in the School of Computing at the University of Leeds. The following objectives were achieved:

Objective 1 and 2: This project successfully obtained personal profiles of supervisors in the School of Computing in University of Leeds through a web scraping tool and with customized data cleansing methods, different versions of dataset obtained, such as a normal version without HTML tags, special characters, and expanded acronyms removed, a version which deleted generic nouns and a version designed for certain vectorization method which removed all subjects and connectives. All those materials were stored in CSV files.

Objectives 3 and 4: This project investigated and evaluated various recommendation approaches and vectorization techniques for developing a supervisor recommendation system. And gives the conclusions of the most applicable and efficient approaches and techniques for this recommendation system.

Objectives 5: This project tested different text generation tools and used them provide a brief introduction for each supervisor, then stored those summaries in CSV files.

Objectives 6: Investigate and evaluate topic modeling techniques for recommendation system and compare with other evaluated recommendation approaches.

Objectives 7: This project designed and deployed a cluster-based recommendation approach that can recommend supervisors with similar research topics or backgrounds and its performance has evaluated.

Overall, this project successfully achieved its objectives and providing deliverables that can be referenced.

The final system, while being a proof of concept, successfully demonstrated the potential of these techniques in improving the supervisor recommendation process. It managed to provide personalized and relevant supervisor recommendations based on user input, generate brief supervisor summaries, and offer additional similar supervisor suggestions, effectively streamlining the supervisor selection process.

In summary, the project successfully achieved its aim of exploring and evaluating various machine learning and natural language processing techniques to build a reliable supervisor recommendation system.

## 6.2 Future Work

The current project remains at the proof-of-concept research stage. Several future expansions can make it become a fully-functional web application, even a general recommendation system serving in multiple areas. There are two main directions of future scope:

Web Application: In the future, the project can be deployed on servers with a Java/Python back-end. A robust data management system will also be designed and deployed, capable of updating and storing information on faculty members across the school. Beyond this, the application can be expanded and deployed to other schools within the university, or even multiple universities. This expansion necessitates the redesign of data scraping and cleaning methods to fit the requirements of different subjects in various schools. Lastly, an essential component of this application is a user-friendly front-end interface. The proposed user interface will resemble a search engine, presenting a list of supervisors with brief introductions after user input. Users can then decide to visit supervisors' individual web pages or directly choose them based on these introductions. Moreover, when a supervisor is selected, a list of similar supervisors will be presented. This familiar interface aims to save time and providing more accurate and informed choices.

Generic Recommendation Systems for Academics: Another possible future direction is to generalize the system into a universal recommendation system that can provide various types of recommendations for different users and scenarios. For example, besides recommending supervisors to students, the system could also recommend potential collaborators to researchers, course suggestions to students, or relevant literature to researchers and students. This would require the system to be able to handle different kinds of data sources and user preferences. Firstly, this system needs to be further enhanced for data cleaning. The current data cleaning method is designed for supervisors in the School of Computing, in the future, data cleaning methods with greater applicability for different content will be designed. Moreover, vectorization methods which are more applicable for long texts, or short texts will also be investigated to handle different types of recommendation tasks respectively.

By implementing these enhancements, the project could represent a comprehensive, user-oriented recommendation system, providing valuable suggestions not only for students seeking supervisor recommendations, but also for the broader academic population including researchers, faculty and students. This system would help increasing efficiency, provide accurate recommendations and foster collaborations.

# List of References

Adomavicius, G. and 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 and data engineering*. **17**(6), pp.734–749.

Al Fararni, K., Aghoutane, B., Riffi, J., Sabri, A. and Yahyaouy, A. 2020. Comparative study on approaches of recommendation systems *In*: *Embedded Systems and Artificial Intelligence*. Singapore: Springer Singapore, pp.753–764.

Ao, J., Wang, R., Zhou, L., Wang, C., Ren, S., Wu, Y., Liu, S., Ko, T., Li, Q., Zhang, Y., Wei, Z., Qian, Y., Li, J. and Wei, F. 2021. SpeechT5: Unified-modal encoder-decoder pre-training for spoken language processing. *arXiv [eess.AS]*. [Online]. Available from: http://arxiv.org/abs/2110.07205.

Blei, D.M., Ng, A.Y. and Jordan, M.I. 2003. Latent Dirichlet Allocation. *Jmlr.org*. [Online]. [Accessed 28 July 2023]. Available from: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf?ref=https://githubhelp.com.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. 2018. BERT: Pre-training of deep bidirectional Transformers for language understanding. *arXiv [cs.CL]*. [Online]. Available from: http://arxiv.org/abs/1810.04805.

Grootendorst, M. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv [cs.CL]*. [Online]. Available from: http://arxiv.org/abs/2203.05794.

Huang, L., Luo, H., Li, S., Wu, F. X., & Wang, J. 2021. Drug–drug similarity measure and its applications, *Briefings in Bioinformatics*, 22(4), bbaa265. doi: https://doi.org/10.1093/bib/bbaa265.

Hastie, T., Tibshirani, R. and Friedman, J. 2009. *The elements of statistical learning*. New York, NY: Springer New York.

Jolliffe, I.T. and Cadima, J. 2016. Principal component analysis: a review and recent developments. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*. **374**(2065), p.20150202.

Le, Q.V. and Mikolov, T. 2014. Distributed representations of sentences and documents. *arXiv [cs.CL]*. [Online]. Available from: http://arxiv.org/abs/1405.4053.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv [cs.CL]*. [Online]. Available from: http://arxiv.org/abs/1910.13461.

Mihalcea, R. and Tarau, P. 2004. TextRank: Bringing Order into Texts. *Aclanthology.org*. [Online]. [Accessed 28 July 2023]. Available from: https://aclanthology.org/W04-3252.pdf.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv [cs.CL]*. [Online]. Available from: http://arxiv.org/abs/1301.3781.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv [cs.LG]*. [Online]. Available from: http://arxiv.org/abs/1910.10683.

Ramos, J. 2003. Using TF-IDF to determine word relevance in document queries. *Psu.edu*. [Online]. [Accessed 28 July 2023]. Available from: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b3bf6373ff41a115197cb5b30e57830c16130c2c.

Rousseeuw, P.J. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*. **20**, pp.53–65.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. 2017. Attention is all you need. *arXiv [cs.CL]*. [Online]. [Accessed 28 July 2023]. Available from: http://arxiv.org/abs/1706.03762.

Verma, V. and Aggarwal, R.K. 2020. A comparative analysis of similarity measures akin to the Jaccard index in collaborative recommendations: empirical and theoretical perspective. *Social network analysis and mining*. **10**(1).

Richardson, L. (2007). Beautiful Soup Documentation. [online] Available at: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ [Accessed 18 Jul. 2023].

OpenAI (2021) 'OpenAI API', OpenAI.

Hugging Face. (2023) Hugging Face – On a mission to solve NLP, one commit at a time. [online] Available at: https://huggingface.co/models [Accessed 19 Jul. 2023].

scikit-learn developers (2021) 'sklearn.feature\_extraction.text.CountVectorizer', scikit-learn 1.3.0. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.CountVectorizer.html [Accessed: 20 July 2023].

scikit-learn developers (2023) Selecting the number of clusters with silhouette analysis on KMeans clustering. Available at: https://scikit-learn.org/stable/auto\_examples/cluster/plot\_kmeans\_silhouette\_analysis.html [Accessed: 20 July 2023].

# Appendix A External Materials

## A.1 GitHub Repository of the Project

https://github.com/tianjianmeng/Final-Project

## A.2 Dataset acquisition source

<https://eps.leeds.ac.uk/computing/stafflist>

## A.3 External Libraries

requests: A Python library for sending HTTP requests. Web page: https://docs.python-requests.org/en/master/

bs4: A Python library for parsing and extracting HTML and XML documents. Web page: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

pandas: A Python library for data analysis and processing, providing a data frame structure similar to R language. Web page: https://pandas.pydata.org/

numpy: A Python library for scientific computing, providing multidimensional arrays and matrix operations. Web page: https://numpy.org/

torch: A Python library for deep learning, based on PyTorch framework, supporting tensor computation and dynamic computation graph. Web page: https://pytorch.org/

scipy: A Python library for scientific computing, providing statistics, optimization, integration, linear algebra, Fourier transform, signal and image processing, ordinary differential equation solving and other functions. Web page: https://www.scipy.org/

scikit-learn: A Python library for machine learning, providing classification, regression, clustering, dimensionality reduction, feature extraction, model selection and other algorithms and tools. Web page: https://scikit-learn.org/stable/

nltk: A Python library for natural language processing, providing lexical, syntactic, semantic analysis, text classification, information extraction, machine translation and other functions. Web page: https://www.nltk.org/

stopwords: A Python library containing stop word lists in multiple languages, which can be used to filter meaningless words. Web page: https://pypi.org/project/stop-words/

CountVectorizer: A Python class for text feature extraction which can convert text into word frequency matrix. Web page: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.CountVectorizer.html

TfidfVectorizer: A Python class for text feature extraction which can convert text into TF-IDF weight matrix. Web page: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.TfidfVectorizer.html

gensim: A Python library for topic modeling and text similarity retrieval, supporting word vectors, LSA, LDA, HDP and other algorithms. Web page: https://radimrehurek.com/gensim/

Word2Vec: A Python class for word vector training which can map words to vectors in high-dimensional space. Web page: https://radimrehurek.com/gensim/models/word2vec.html’

## A.4 External codes (silhouette analysis) learned from scikit-learn.org

<https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html>

# Appendix B Ethical Issues Addressed

This project involved collecting and analysing data from public webpages of university faculty members. As such, it necessitated careful consideration of ethical issues surrounding privacy, consent, and appropriate data use.

## B.1 Privacy and Consent

The project exclusively used public information on faculty webpages that the university and faculty members intentionally make accessible online. No private data sources were accessed. This mitigates privacy concerns. The project complied with university data usage regulations by only using public data for legitimate academic purposes.

## B.2 Data Use

This project only stores public faculty member information in a public GitHub repository. Beyond that, any other data generated during the experiment, such as user input, is not stored.

## B.3 Equity and Bias

Natural language processing techniques may introduce issues of bias. This may result in some supervisors being more likely to be recommended than others, such as supervisors with much text are more likely to be recommended than supervisors with less text in some cases. This may be unfair. To avoid this, it is necessary to let supervisors write profiles in a certain style and have a length limit, or improvements to the existing algorithms can ultimately alleviate this unfair situation.