Personalized Research Assistant

Team Quest

Group Members

145018N	Dilrukshi S.A.P.S.
145030T	Jayakodi S.M.
145061M	Premarathne A.G.M.I.
145068P	Rathnayake K.V.R.A.M.

Supervised by:

Dr. (Mrs.) Thushari Silva

Dr. (Mrs.) Thanuja Sandanayake

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfilment of the requirements of the Honours Degree of Bachelor of Science in Information Technology & Management.

February 2019

Declaration

We declare that this thesis is our own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of Students	**Signature**
Dilrukshi S.A.P.S.	.....
Jayakodi S.M.	.....
Premarathne A.G.M.I.	.....
Rathnayake K.V.R.A.M.	.....

Date: .....

Supervised by:

Names of the supervisors	Signature of Supervisors
Dr. A.T.P. Silva, Senior lecturer, Department of Computatuional Mathematics	.....
Dr. Thanuja Sandanayake Senior Lecturer, Department of Interdisciplinary Studies	.....

Date: .....

Abstract

The world is full of innovations and those innovations mainly based on researches. In the past years scientific research papers has increased because people are always trying to do something new to go with the flow of technology. There are students, undergraduates, graduates and post graduates who are willing to explore for new things and publish their new findings as research papers. With the advanced development of technology, the researchers also find easy ways to get help to do their researches and publish them. In here the proposed system based of IEEE and Google Scholar Personalized Research Assistant could be the best matching supporter for the researchers. When doing a research there is somewhat hard to find the best matching collaborators for further work because usually research papers are not published individually, as a solution for that we give the platform for the user to find the best matching collaborators based on the expert area and connectivity. As well as we give the opportunity for the researcher to get updated about the upcoming conferences related to his specified areas and we are giving recommendations about where is best to publish the research papers as well. All of these co-related things are based on an automatically generated user profile which gets details from the Google Scholar account of the user. Latent Dirichlet Allocation (LDA), web crawling, filtering techniques, Data mining techniques and both supervised and unsupervised machine learning are used in the successful implementation of the system. This is a very user friendly system because the researchers find a single platform to get all the supports for his research.

Acknowledgement

We would like to offer our gratitude to our supervisors, Dr. (Mrs.) A.T.P. Silva and Dr. (Mrs.) Thanuja Sandanayake for the great help and support for the whole period of time throughout the project. There were many people behind of the success who helped us a lot. We would like to thank our seniors who gave us guidance from their experience. We kindly commemorate the nonacademic staff of the Faculty of Information Technology for the assistance mainly with materials and lab facilities. We would like to thank our family members as well for the great understanding and support for this whole period of time. Finally our compliment goes to our colleagues who helped us in many ways in the development of the project. It would not be possible without these supports so that we remember everyone who supported even with a single word more gratefully.

Table of Content

List of Figures

Chapter 01

Introduction

1.1 Introduction

Because of the rapid increment of technology, the huge amount of data extremely increases in the Web by recent trends. Because of this lots of research papers are being published per year. Because of that that is not easy for the researchers to find relevant specific articles, find collaborators to work with them and find the most relevant platform to publish their research papers kind of things. When it comes to the new bees for the research writing such as undergraduates and the students doing higher studies find it hard and time consuming task finding relevant resources when writing a research paper and publishing that.

The digital libraries like Google Scholar, IEEE and ACM provide the researchers keyword based recommendations but, when considering accuracy it is low and by following those recommendations the possibility of missing the most relevant ones is high[1]. Google Scholar provides the collaborators related to a particular researcher but searching for the profile is essential to find those [5]. This is disadvantageous for the new ones as well as the experienced researchers to go for several sites to search for relevant details.

To address those problems, this research provides a single platform for a particular researcher who logs into the system to find the collaborators, connections and to find the most relevant platform to publish the research by auto generating user profile with his specialized field.

This solution has four main features; (a) auto generating user profile based on the abstracts written by the researcher (b) social network analysis (c) recommending collaborators for the user (d) Recommending the best upcoming conferences to publish the paper. When considering the previously done systems they have followed content based filtering, collaborative filtering and hybrid approaches and this research provides a developed system with several facilities combined together.

1.2 Background & Motivation

For the researchers there are several sites which support them to find out the related platforms for publications, supervisors and research opportunities. These search terms return a list of matched details with site information. The researchers have to evaluate those results themselves. To find all the relevant information they have to search for many sites. There is not a single system that provides personalized recommendation for related platforms for publications, supervisors and research opportunities. The existing systems have issues in measuring quality of the recommended facts, relevance to the user profile and connectivity.

1.3 Problem in Brief

In the existing systems that provide support for the researchers, but there is not a system available that gets the details of the users and then recommends most supportive and best materials. The users have to go through many sites, tools and research papers to find the relevant information. Also, user have to provide details and information manually which required to as above mentioned systems. Therefore, searching that information is difficult and time-consuming process. In details, the researchers have to put effort to find supervisors, platform of publication and research opportunities. When publishing the research papers that encounters several problems. Publication may be return due to issues related with the aim and scope of research paper. There are three types of return as fully return, return with small issues and return with major issues.

1.4 Aim & Objectives

1.4.1 Aim

The aim of this proposal is to develop a profile-based product that provides a single platform for the researchers to get all the services they find from several products into one.

1.4.2 Objectives

1. To auto generate a profile for every researcher based on their personal preferences, skills, research areas when user login to the system.

2. To make recommendations for platform of publications to the users based on their unpublished research papers.

3. To make research opportunities recommendation to the users based on their personal favors in researches by referring the profile

4. To find and recommend available supervisors based on the profile.

1.5 Personalized Research Assistant (Proposed Solution)

To defend the above mentioned problem, a personalized research recommendation system is proposed which has the ability to provide the researcher the collaborators, social network analysis and best platforms to publish the research paper by auto generating a profile. This system consists of four sub modules as follows,

- ✓ *Automatic profile generation*
- ✓ *Social network analysis module*
- ✓ *Collaborators recommendation module*
- ✓ *Recommendation of journals for manuscript*

1.6 Resource Requirement/ Availability

There are software as well as hardware requirements needed for this research such as research papers, datasets, sites and user profiles. Research Gate was the main site which searched for the profiles. The research papers were available in IEEE Xplore, Research Gate and ACM for further details. Whole project is based of python programming and several techniques are being used.

1.7 Summary

In this chapter, a brief introduction of the project is provided including aim and objectives. Main modules are described here as well. Next chapter will provide a review on others work.

Chapter 02

Related Work

2.1 Introduction

This chapter will discuss about the existing systems which are done by others to overcome the same problem of the research. Recommending systems have been a major way of providing ease to the users so that there are so many systems and personalized research articles for the same problem domain. Because of the growth of technology and as an area that regularly get updated many research articles and related systems can be found in the digital libraries as Google Scholar, IEEE and ACM. This chapter includes their approaches, technologies used, advantages and limitations compared to the Personalized Research Assistant in detail for each sub module separately.

2.2 Common Systems

2.2.1 ScolarMate

ScholarMate is a platform for sharing publications and disseminating research outputs. Using techniques of big data analytics, researchers can receive recommendations on relevant opportunities based on their profiles. Registered users can easily share their publications with their peers so as to create impact and increase citations using the research social network platform [2]. ScholarMate provides personalized services for status updates, research profiles, publications, and funding opportunities etc., it allows researchers to create groups for their projects and courses.

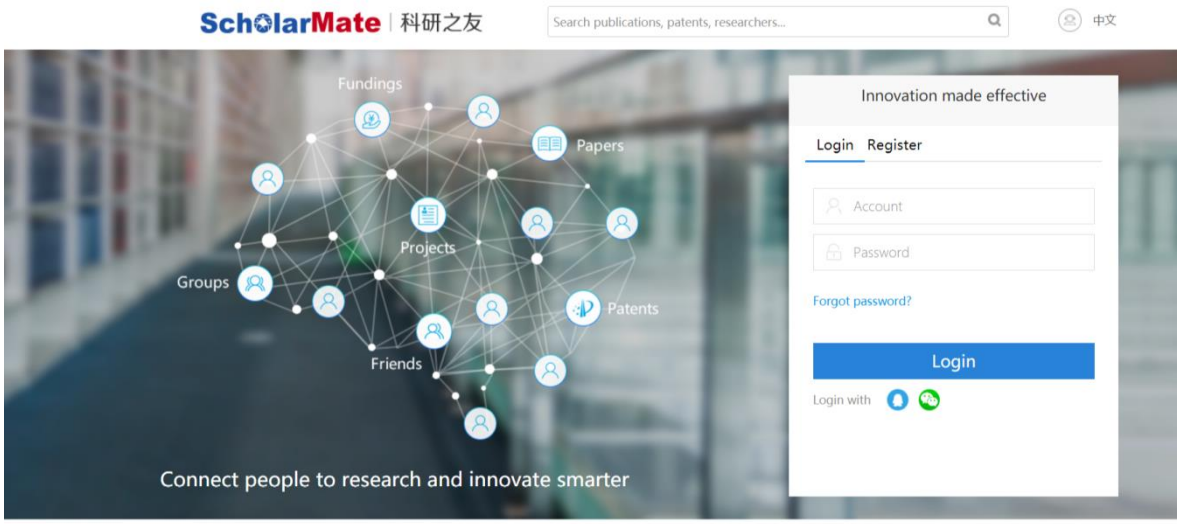


Figure 2.1: Interface of the ScholarMate

2.2.2 Jane

Jane helps to find journals to publish recently written papers, helps to find relevant articles to cite in the written paper for researchers and it helps to find reviewers for a particular paper[3]. User should enter the title and the abstract in the given text boxes and then Jane provides its service. Apart from that Jane provides the Keyword search as well.

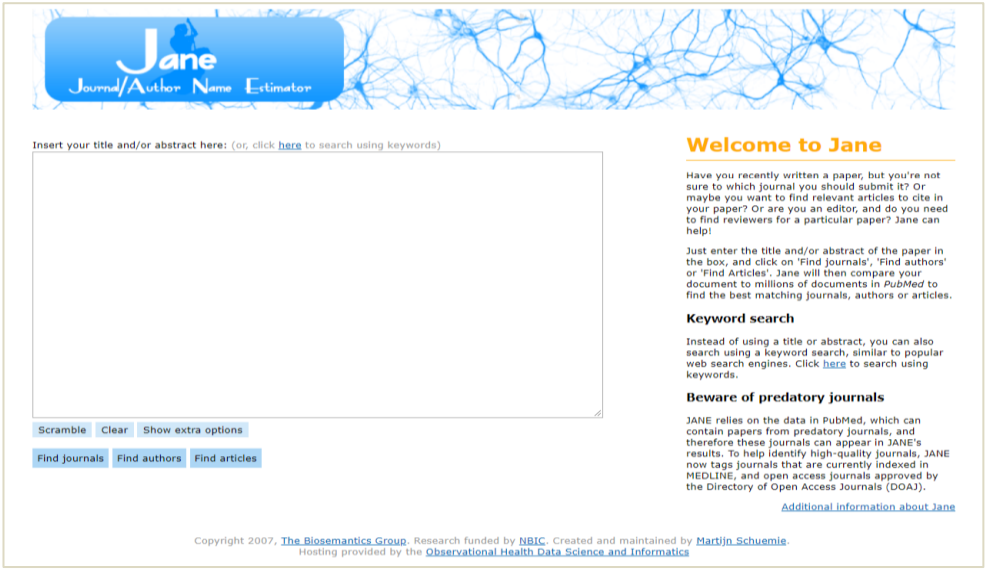


Figure 2.2: Jane interface

2.2.3 Springer journal suggester

Springer journal suggester helps to select the journal that suits for given details by the researcher from over 2,600 Springer publications. It uses semantic technology to find quickly the suitable Springer journal that is right for given information. First user should enter the abstract, description of the research, or a sample text and the Springer Journal Selector provides a list of relevant journals [4]. Then user can refine the results based on requirements for Impact Factor or publishing model, including an option to match to journals that are fully open access or have open access options.

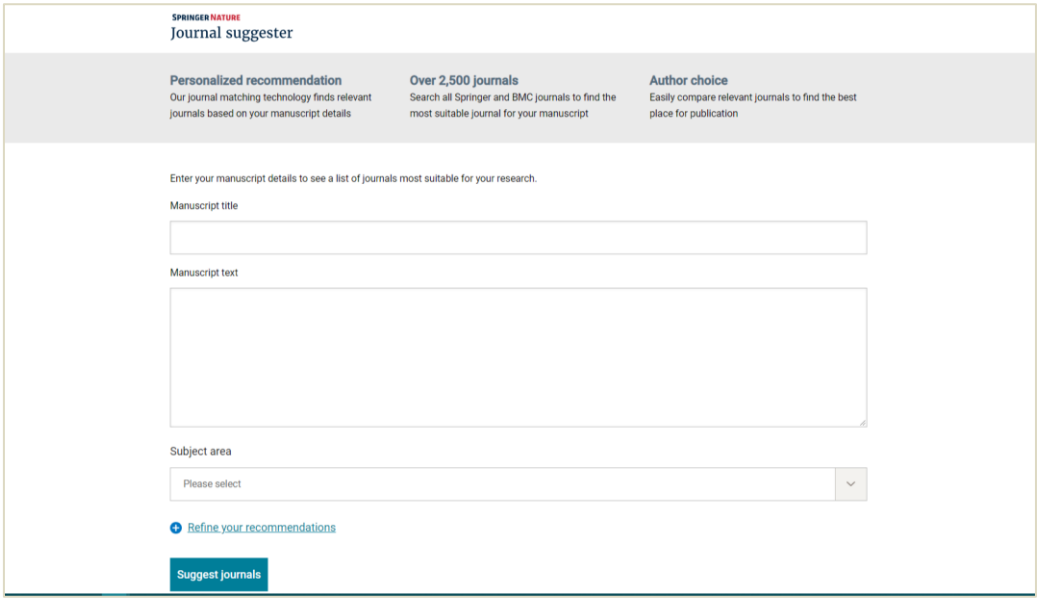


Figure 2.3: Springer journal suggester

2.3 Profiling module

Recommendation systems are the modern way of filtering information. This makes ease for the users to get a large, various items for their interests. Profiling is the base of such recommendation systems because it causes for guiding an individual in personalized way to interesting or useful objects in a large space of possible options [6]. But this is not an easy task to give the user information according to their individual interests by classifying and ranking so that profiling is a most important part. There are systems and researches done based on profiling and those will be discussed in details with the mostly used techniques are support vector machine and naive bayes algorithm.

- User Profile Based Personalized Research Paper Recommendation System

User Profile Based Personalized Research Paper Recommendation System (by K wanghee Hong, Hocheol Jeon, Changho Jeon) is a similar approach for the proposed system which is done the purpose of accuracy of finding research papers based on user profile and to apply the system for practical use. This provides a high level solution for the lack of user profile information sensitivity by reflecting the profile information. In this research the topic of the research papers of the users are used to store preference information. [5]

Keyword extraction is done by removing unnecessary characters by building up an algorithm as follows.

Algorithm 1 Keyword Extraction

```
1:  function KeywordExtraction(content)
2:  // preprocessing the content of a paper to get 'Keywords'
    section of the paper
3:  key_content ← getKeywordsSection(content);
4:  // removing reserved keyword 'keywords' from keyword
    section
5:  key_content ← removeKeyword(key_content);
6:  // Tokenizing by ';' or ','
7:      while exist token do // repeat until there are no more
    tokens
8:          key ← getToken(key_content); // tokenize from
    key_content
9:          Keywords[i] ← key; // add key to Keywords at i
10:         i++; // increase index i
11:     end while
12:  return Keywords;
13: end function
```


Figure 2.4: Keyword extraction algorithm

In this research XML format is used to store the profile data because of the ease of tracing information. In every single click of research papers the profile is being updated as well and finally the top 50 research papers are recommended to the user based on the updating user profile.

For these tasks, tokenization, web crawling, keyword matching and filtering techniques are used.

- Personalization of Web Search Results Based on User Profiling

Another related research is the Personalization of Web Search Results Based on User Profiling (by Snigdha Gupta,Saral Jain andMohammad Kazi). **[6]** In this research a personalized learning based dynamic database is being created to reflect his references. Among various types of profiling, this research has used the content based profiling technique which captures user information by different aspects. In content based profiling this system has user term based profiling, category based profiling and link based profiling. Term based profiling has used the data extracted from scanned documents and TF/IDF has being used here and when a term is available in the document a weight is being added.

Term frequency model output is below,

```
<domain_info>
<domain name="Artificial Intelligence">
<topic name="Information Retrieval" wDt="0.4827586206896552">
<keyword name="Learning to Rank" wtk="0.03225806451612903"/>
<keyword name="Benchmark Datasets" wtk="0.03225806451612903"/>
<keyword name="MIR" wtk="0.03225806451612903"/>
<keyword name="matching" wtk="0.03225806451612903"/>
<keyword name="indexin" wtk="0.03225806451612903"/>
<keyword name="Information retrieval" wtk="0.1935483870967742"/>
<keyword name="Semantic Web" wtk="0.03225806451612903"/>
<keyword name="DAML+OIL" wtk="0.03225806451612903"/>
<keyword name="OWL" wtk="0.03225806451612903"/>
<keyword name="Linear discriminant analysis" wtk="0.03225806451612903"/>
<keyword name="High dimensional data" wtk="0.03225806451612903"/>
<keyword name="Simultaneous diagonalization" wtk="0.03225806451612903"/>
<keyword name="Face recognition" wtk="0.03225806451612903"/>
<keyword name="Peer-to-Peer Information Retrieval (P2P-IR)" wtk="0.03225806451612903"/>
<keyword name="architecture" wtk="0.03225806451612903"/>
<keyword name="key-based routing (KBR)" wtk="0.03225806451612903"/>
<keyword name="P2P web search" wtk="0.03225806451612903"/>
<keyword name="information" wtk="0.03225806451612903"/>
<keyword name="modern information retrieval" wtk="0.03225806451612903"/>
<keyword name="brief overview" wtk="0.03225806451612903"/>
<keyword name="association thesauru" wtk="0.03225806451612903"/>
<keyword name="semantic representation" wtk="0.03225806451612903"/>
<keyword name="information retrieval research" wtk="0.03225806451612903"/>
<keyword name="mml query" wtk="0.03225806451612903"/>
<keyword name="two rapidly consecutive stimuli" wtk="0.03225806451612903"/>
<keyword name="semantic web doe it exist" wtk="0.03225806451612903"/>
</topic>
<topic name="Focused Crawler" wDt="0.2413793103448276">
```

Figure 2.5: Term frequency model output

Same as the previously mentioned term, category based profiling makes categories along with the corresponding weights and then link based profiling builds links among the categories. Context analysis method and TF/IDF have being used in this research for profiling.

2.4 Recommendation of journals and conferences for manuscripts.

This module will be provided Recommendation of journals and conferences for manuscript. User can upload a research paper to system then the system should be identified main features of that paper. Especially this module should extract paper's title, abstract, keywords, and references. To archive that progress, I use some python libries.eg:-pdfminner, pdf2txt.Next process is fetching user profile data. When publishing a research paper has to consider more things. Researcher has to choose suitable place. If it is wrong place, paper will be rejected. There for researchers have to consider more things when they publishing something. Also I hope to increase quality of recommendation. Therefore I use user profile data for making connective and productive manner.eg:-skill and qualifications, interesting, experience. Next creates a common model using both information categories. Next thing is selecting relevance resources (journal/conference) via Web. I hope to extract some necessary data from web such as similar topics published places, journal's aim and scope, previous published paper's ratings. I expect to use beautifulsoup and web crawler for extract these kinds of data from relevant web sites. After extract I hope to create a profile and it is based on extracted data from journals or conferences. I hope to create few common formats for archiving data analyze. Some journals are allowed to create multiple themes. But some journals are unique. There for I hope to create some common formats. Finally it will create profile of journals or conferences.

Next part is scheduling to match article profile(model) and journal profile I have to use matching algorithms for archiving relevance, quality, connectivity, productivity.I decide to use below machine learning techniques such as K-Means Clustering ,MeanShift Clustering ,GMM, Fuzzy c-mean. Known that there are lots of journals/conferences outside. Researchers have to choose suitable place for publishing their works. This module will recommend suitable place for user.

2.5 The connectivity detection module

There are several social network analysis related projects are available. Link prediction in networks has been an important topic since the emergence of online social networks. Here describe about the most related work with this module. This work consider about the problem of predicting future co-author relationships between existing authors in a heterogeneous bibliographic network and using heterogeneous topological features. This is different from the traditional co-author network setting. Because a heterogeneous bibliographic network is considered, this contains multiple types of objects, such as authors, venues, topics and papers.

In here, they use the DBLP bibliographic network as an example of heterogeneous bibliographic networks. The DBLP bibliographic dataset with citation information provided by consists of rich information for publications, such as the authors, venues, titles and so on.

Schema for DBLP Bibliographic Network is as follows,

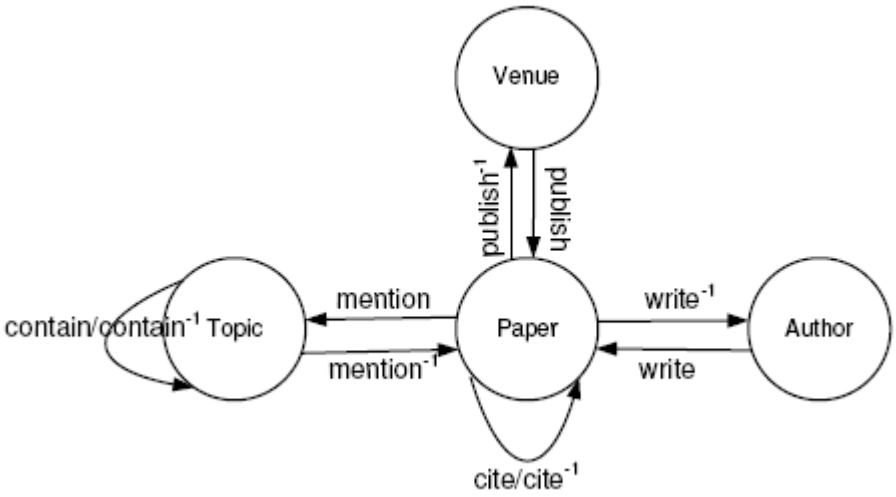


Figure 2.6: Schema for DBLP Bibliographic Network

The supervised framework is used to obtain the Co-Author Relationship Prediction task. The proposed approach is the Path Predict model to address this problem, which first defines Meta path based. Experiments on the DBLP bibliographic network show that by considering heterogeneous topological features, the relationship prediction accuracy can be significantly improved, and the model using hybrid features that have combined different meta paths and different measures gives the best overall

performance. Finally, topological features in such networks, and then builds logistic regression-based co-authorship prediction model.

2.6 Collaborators Recommending Module

Most of the recommendation systems based on two approaches as people to people recommendation and object to people recommendation. Recommending collaborators belong to the people to people recommendation approach. There are such scientific recommender systems such as journal, article and supervisor and collaborators and for those there are mostly used approaches such as content based, collaborative filtering and hybrid.

- Content Based Filtering

  Content based filtering compares the profile of the user with the content simply such as the words in a document. This technique is mostly used with text documents and most of the recommending systems have used vector space model and latent semantic indexing for content based recommender systems.

  Vector space model can be identified as the way of representing text documents in an algebraic manner.

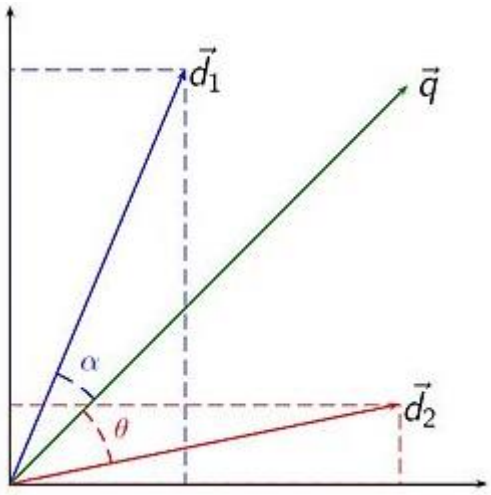


Figure 2.7: Graphical representation of vector space model

Documents are represented as vectors in this module and for separate terms there are separate dimensions. A term can be identified as keywords or phrases in here TF/IDF technique are mostly used. When comparing documents with queries vector space model is used.

- Collaborative Filtering

In collaborative filtering technique most of the related projects have used the Pearson correlation algorithm [24] [25]. This implements the relationship between variables with r. This draws a line between those variables and this can be identified as the best matching. Then it measures how far the data points are with respect to the best patching point [45].

$$r_{xy} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sqrt{(\sum x_i^2 - n\bar{x}^2)}\sqrt{(\sum y_i^2 - n\bar{y}^2)}}.$$

Pearson correlation equation

There are three main types of correlations between variables can be identified using this as positive correlation, negative correlation and no correlation.

Types of correlations can be identified in Pearson's correlation coefficient

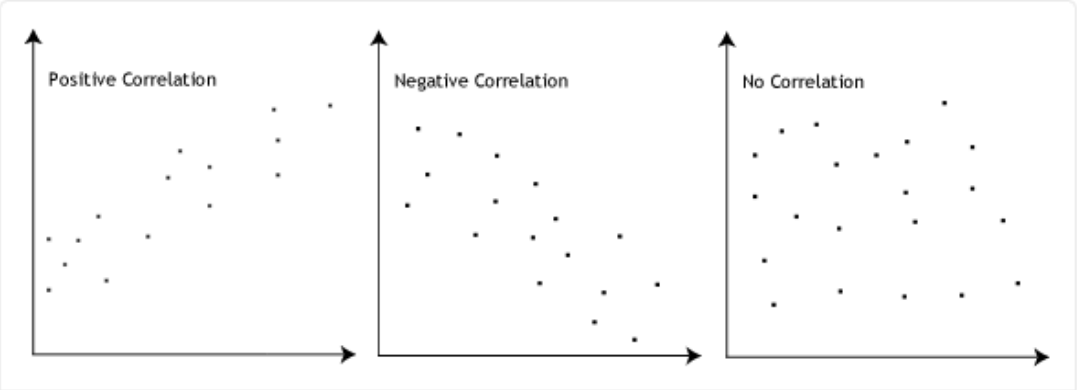


Figure 2.8: Pearson's correlation coefficient types

Keyword correlation matrix is another approach used to measure the correlation between variables. This is a data summarizing technique as well. This technique is used to get a large set of data summarized while identifying the patterns between them.

- Hybrid Filtering

To overcome the shortcomings of each separate module, hybrid filtering method is used by combining two or more techniques together. In most of the recommendation systems hybrid approach is used by combining techniques such as content based, collaborative filtering and knowledge base filtering.

- In research analytics framework-supported recommendation approach for supervisor selection research (by Mingyu Zhang, Jian Ma, Zhiying Liu, Jianshan Sun and Thushari Silva, 2016),a novel student-centric method is proposed for finding and recommending appropriate supervisors for new students. In this research the quality, connectivity and relevance dimensions are computed and finally three dimensional scores are aggregated by employing by employing the entropy method (Xu, 2004). Based on the aggregated results, the final ranking list was obtained. Direct keyword matching technique is used to match the supervisors with the students in this research and hybrid approach is used for system implementation [47].
- Learning Collaborative Filtering and Its Application to People to People Recommendation in Social Networks (by Xiongcai Cai, Michael Bain, Alfred Krzywicki, Wayne Wobcke, Yang Sok Kim, Paul Compton and Ashesh Mahidadia) [44] has used Model Based and Memory Based approaches of collaborative filtering for recommendations in Social networks. The recommendation part is done by using CF+ modeling in user based collaborative filtering [41].
- Recommender System Using Clustering Based On Collaborative Filtering Approach (by Jyoti Pandey, Prof. M. R. Patil, 2015) idea of object typicality from cognitive psychology borrowed and proposed a typicality-based CF recommendation approach named OTCO. In here Pearson correlation coefficient is used to measure similarity between items and then top k neighbors are selected using k- means clustering algorithm [18].

Same as the above mentioned ones several various methods have been proposed for people to people recommendation such as Analytical Network Process (ANP), Multi Criteria Decision Making (MCDM) and Analytical Hierarchy Process (AHP). However each of those approaches have limitations and from the proposed approach, overcoming these limitations is expected [25] [20].

2.7 Summary

In this module brief descriptions of the related work of the whole system and each of the modules are being given and the next chapter will be discussed about the technologies used for the implementation of the system with details.

Chapter 03

Technology adapted

3.1 Introduction

In the previous chapter, literature review was discussed. This chapter gives detailed information about the technologies which were used when implementing the solution. The system based on python and other technologies were used as well for the successful implementation of the project. Free and open source libraries were used for the support to extract data and build machine learning algorithms. Python was selected because of the ease of use and several libraries were used for hard parts.

3.2 Python

Python language is used in the system because it is easy and can understand quickly. It is a general purpose programming language. Python is a scripting language like PHP, Perl, Ruby and so much more. It can be used for web programming (Django, Web2py, Zope, Google App Engine, and much more). But it also can be used for desktop applications (Blender 3D, or even for games pygame). Python can also be translated into binary code like java. The syntax and the structure of Python language is easy to learn and implement and intuitive too. Here used the version 3.6. It is very supportive for many libraries.

3.3 Python Libraries

3.3.1 NLTK

The Natural Language Toolkit is an open source library for the Python programming language originally written by Steven Bird, Edward Loper and Ewan Klein for use in development and education.

It comes with a hands-on guide that introduces topics in computational linguistics as well as programming fundamentals for Python which makes it suitable for linguists who have no deep knowledge in programming, engineers and researchers that need to delve into computational linguistics, students and educators.

3.3.2 Pandas

In computer programming, the pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

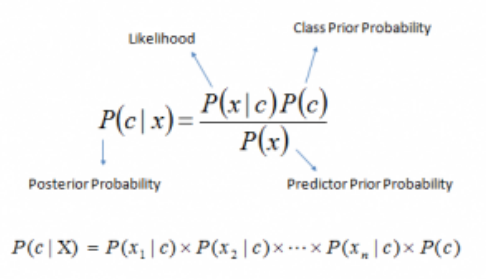
3.3.3 Sklearn

Software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. In Sklearn, following classes were used**.**

- **Naïve Bayes Classifier**

  Naive Bayes algorithm is used for classification. The simplest way of describing naive Bayes is that this assumes that a particular feature in a particular class in not relate to any other features in the class.

  As an example if a fruit is round shaped, orange color and about two inches of time even if the features are together or identified separately it contributes to assume that the fruit is an orange. This is said to be ‘Naive’. Naive Bayes algorithm is helpful for varse data sets but this is being called as a outperform algorithm because of the simplicity.



Above,

- $P(c|x)$ is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

Figure 3.1: Naive Bayes equation

- **Support Vector Machine**

Support Vector Machine can be identified as a supervised machine learning algorithm which is being used mostly for classification as well as regression. The algorithm is used to plot the data items in an n- dimensional space one by one with the values of coordinates. Then the classification is done by finding the hyper-plane which differentiate classes.

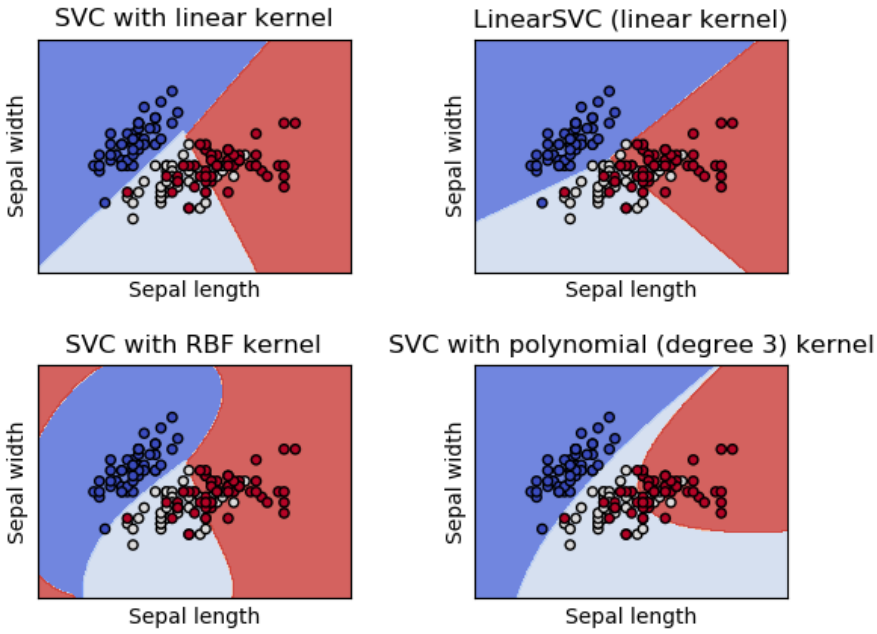


Figure 3.2: Graphical representation of SVM

3.3.4 PDFMiner

PDFMiner is a tool for extracting information from PDF documents. Unlike other PDF-related tools, it focuses entirely on getting and analyzing text data. PDFMiner allows one to obtain the exact location of text in a page, as well as other information such as fonts or lines. It includes a PDF converter that can transform PDF files into other text formats (such as HTML). It has an extensible PDF parser that can be used for other purposes than text analysis.

3.3.5 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

3.3.6 Beautiful Soup

This is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup). It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping. (https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser))

3.3.7 Scholar

Scholar.py is a Python module that implements a querier and parser for Google Scholar's output. Its classes can be used independently, but it can also be invoked as a command-line tool.

3.3.8 Selenium

Selenium is a portable framework for testing web applications. Selenium provides a playback (formerly also recording) tool for authoring functional tests without the need to learn a test scripting language (Selenium IDE). It also provides a test domain-specific language (Selenese) to write tests in a number of popular programming languages, including C#, Groovy, Java, Perl, PHP, Python, Ruby and Scala. The tests can then run against most modern web browsers. Selenium deploys on Windows, Linux, and macOS platforms.

3.3.9 csv

The **csv** module gives the **Python** programmer the ability to parse **CSV** (Comma Separated Values) files. A **CSV** file is a human readable text file where each line has a number of fields, separated by commas or some other delimiter. You can think of each line as a row and each field as a column. (https://dzone.com/articles/python-101-reading-and-writing)

Python has a shallow learning curve, its syntax and semantics are transparent, and it has good string- handling functionality. As an interpreted language, Python facilitates interactive exploration. As an object-oriented language, Python permits data and methods to be encapsulated and reused easily. As a dynamic language, Python permits 9 attributes to be added to objects on the fly, and permits variables to be typed dynamically, facilitating rapid development. Python comes with an extensive standard library, including components for graphical programming, numerical processing, and web connectivity. Python is heavily used in industry, scientific research, and education around the world. Python is often praised for the way it facilitates productivity, quality, and maintainability of software. Python support machine learning via various libraries developed by researchers around the globe. It all facilitates NLP functions more efficiently and effectively in relation to other languages available, which is vital to this research. Apart from this Sentiment Analysis and Machine Learning were used.

3.3.10 FuzzyWuzzy

FuzzyWuzzy is a string matching algorithm which is used to match the names of the users to find the common set of users in the system for recommendation.

3.3.11 Neo4j

Neo4j is a graph database management system developed by Neo4j, Inc. Here, use the Neo4j for storing data that has many interconnecting relationships and for visualizing as a network for get connectivity of each author. Neo4j stores and presents data in the form of a graph.

Neo4j is based on Cypher query language. Data is represented by nodes and relationships between those nodes. Here also used apoc library for import JSON data

to neo4j browser. Graph algorithm library is used to compute metrics for graphs, nodes, or relationships.

3.3.12 Cypher

Cypher is Neo4j's graph query language. It allows users to store and retrieve data from the graph database and that allows for expressive and efficient querying and updating of the graph. However, Cypher borrows its structure from SQL queries are built up using various clauses. Cypher's syntax gives a visual and logical way to match patterns of nodes and relationships in the graph.

3.3.13 Neo4j Browser

The Neo4j browser is a graphical user interface (GUI) that can be run through a web browser. It can be used for adding data, running queries, creating relationships, and more tasks. It also provides an easy way to visualize the data in the database.

3.3.14 APOC library

The APOC library consists of many procedures and functions that can't be easily expressed in Cypher itself. This library helps to many different tasks in areas like data integration, graph algorithms or data conversion. Overview in the documentation and detailed in subsequent sections are included all the listed procedures. Here APOC library is used to import JSON data

3.3.15 Py2neo

Py2neo is a client library and comprehensive toolkit for working with Neo4j from within Python applications and from the command line. It has been carefully designed to be easy and intuitive to use.

3.4 Technologies

3.4.1 Sentiment Analysis

Sentiment Analysis (SA), or Opinion Mining (OM) is the use of NLP, text analysis and computational linguistics to identify and extract information (or features) from texts.

3.4.2 Machine learning

Machine learning algorithms in recommender systems are typically classified into two categories content based and collaborative filtering methods although modern

recommenders combine both approaches. Supervised and unsupervised learning algorithms were used here from following items.

3.4.3 MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling. Here, use Neo4j Doc Manager for mongo-connector to facilitate synchronizing data from MongoDB to a Neo4j instance.

3.5 Preprocessing Techniques

There are many basic concepts of text classification that have to use implement this system.

3.5.1 Tokenization

This is the concept include the process of chopping sentences into smaller pieces. Electronic content is a direct grouping of images (characters or words or expressions). Normally, before any genuine content preparing is to be done, content should be sectioned into semantic units, for example, words, accentuation, numbers, alphanumeric, and so on. This procedure is called tokenization. e.g. - I'm a Data Science enthusiast and studying and researching on the fields of Data Science, Machine Learning, Artificial Intelligence & Big Data 11 [I'm] [a] [Data] [Science] [enthusiast] [and] [studying] [and] [researching] [on] [the] [fields] [of] [Data] [Science] [,] [Machine Learning] [,] [Artificial] [Intelligence] [&] [Big] [Data].

3.5.2 Stop word removal process

When working with data mining applications, frequently heard about the expression "stop words" or "stop word list". Stop words are fundamentally an arrangement of normally utilized words in any dialect, not simply English the reason why stop words are critical to many applications is that, if we remove the words that are very commonly used in a given language, Can focus on the important words instead [11]. Usually this process will be started after the tokenization and it removes unwanted words that are including in the tokens in this stage.

e.g. - [I'm] [a] [Data] [Science] [enthusiast] [and] [studying] [and] [researching] [on] [the] [fields] [of] [Data] [Science] [,] [Machine Learning] [,] [Artificial] [Intelligence] [&] [Big] [Data].

Apart from using this stop word technique, created own stop word array.

3.5.3 Lemmatization

For grammatical reasons, reports are going to use different forms of a word. Additionally, there are groups of derivationally related words with same meanings. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set. So, the objective of lemmatization is to reduce inflectional structures and some of the time derivationally related types of a word to a typical base frame.

3.5.4 TF/ IDF

TF-IDF, which stands for term frequency—inverse document frequency**,** is a scoring measure widely used in information retrieval (IR) or summarization. TF-IDF is intended to reflect how relevant a term is in a given document.

The intuition behind it is that if a word occurs multiple times in a document, should boost its relevance as it should be more meaningful than other words that appear fewer times (TF). At the same time, if a word occurs many times in a document but also along many other documents, maybe it is because this word is just a frequent word; not because it was relevant or meaningful (IDF).

3.6 Summary

This chapter described what technologies are being adapted to implement the solution. Each technology will be used to develop one or more modules in the application. The next chapter will explain how the development process is started and how these technologies are being used.

Chapter 04

Our Approach

4.1 Introduction

Previous chapter discussed about the technologies adapted in this system. This chapter describes the approach towards this study. The proposed solution can be interpreted in terms of inputs, output, users and processes.

4.2 Profiling Module

This module can be identified as the base of the system. All the other three modules based on the output given by the profiling module. Profiling module of this system differs from most of the similar systems in case of auto generated profile with the subject area as well. To generate the user profile, first collected the abstracts from Google scholar and IEEE API.

Supervised Machine learning algorithms named Support Vector Machines, Naïve Bayes, Decision Tree classifier, Random Forest classifier, KNN were used to create different models as first approach. In this approach, 3 different methods also used to find the most accurate method. Those methods as follows.

1) As sum of each array length into % [using probability based model]
2) As length of all abstracts into % [using probability based model]
3) As length of all abstracts into % [using class classification model]

First method is predicting the abstract one by one to relevant class. Then each category is being divided from total abstracts and calculates the user interest as a percentage.

As an example, if the abstracts collected as above belong to the predefined classes as,

ML - 23/100
BD - 05/100
DM - 32/100
CV - 0/100
BIO - 05/100
AI - 35/100

Then the final result will be 23% for Machine Learning, 5% for Big Data, 32% for Data Mining, 0% for Computer vision, 5% for Bioinformatics and 35% for Artificial Intelligent.

Second method is getting the abstracts one by one and considers the probability of given abstract for all 6 classes separately. After that if the probability is greater than or equal to 0.5, abstract will be added to that class. In this method, if the abstract has the probability of more than 0.5 for multiclass, that abstract will be added for all those classes.

After getting the all abstracts, next the size of each class is being taken and makes a denominator by getting sum of all those values. After that divide each class size from that denominator and get the value as the percentage. Results getting from this method are more accurate than the first method.

As an example, if the sizes of classes as follows,

ML – 20

BD – 12

DM – 8

CV – 4

BIO – 22

AI – 14

Then the denominator is 20+12+8+4+22+14 = 80

Then the user interests can be calculate as,

ML = (20/80)*100 = 25%

BD = (12/80)*100 = 15%

DM = (8/80)*100 = 10%

CV = (4/80)*100 = 5%

BIO = (22/80)*100 = 27.5%

AI = (14/80)*100 = 17.5%

Third method is Topic modeling using Latent Semantic Analysis and it will be explained future in few paragraphs.

As the second approach, Latent Semantic Analysis model was used. Latent Semantic Analysis is an approach in topic modeling. The results of this were collected separately to compare with the three methods in the first approach. In this approach word vectors are being created for the selected six areas, these word vectors contain

the mostly used words in each area. In here the word count was changed in various amounts and the results were collected for further work. When a new user is logged into the system, the publications of that user are being get and each abstract of each publication and each of those abstracts are being added to the vectors one by one and the most scored vector from each of those is related to that user.

4.3 Recommendation of journals for manuscript

The one of the major issues that researcher faces when dealing with research field is finding appropriate journal or conference for publication. Due to the novelty to the field and limited knowledge are some of the reasons for that fact. Normally researcher search places to publication through the studying guidelines manually. When publishing manuscript in a journal there are some standards and limitations. Before accepting publication particular journal consider about alignment of the aim and the scope of journal with the manuscript. So researchers have to consider and comply with those standards. The implementation is done in few approaches to suggest suitable journal for manuscript. The first approach is extracting manuscript information and detecting the relevant area. Second approach is making a model of manuscript data and user profile data. Third one is making a model of journal profile.it has included relevant rate of aim and scope of journal, impact factor, acceptance rate and finally it will be provided recommendation with high quality and accuracy than manual process.

4.4 The connectivity detection module

Social network analysis examines the structure of relationships between social entities. These entities are often persons, but may also be organizations, groups, scholarly publications, nation states. Basically, the implementation is done in two main approaches to detect connectivity between researchers. The first approach is Co-authorship networks, in which two researchers are considered connected if they have co-authored one or more scientific papers. This is one of the most comprehensive study in a social network. Co-authorship network is built containing one author to reduce the complexity of the network with the purpose of recognition. Co-authors and co-authors of co-authors are also determined based on that limited research paper titles. The second approach is done using the same theories but a different flow with the researcher Affiliation.

Approach 1 - Co-author network

Here this approach visualizes the researcher co-authorship network which is built from the Google scholar data and suggest the strongest connected co-author and the strongest connected co-authors of co-authors. This network behavior is predicted new collaborators to a researcher to do further research works. The input of this module is author name and the research paper title and related co-authors and co-authors of co-authors names for each and every title.

In comparison to the citation network, co-authorship implies a much stronger social bond, since it is likely that a couple of researchers who have co-authored a paper together are personally known to each other. Therefore this approach is suggested that co-authors of co-authors who do not work as collaborators with the particular author but work with the co-author. The co-author is the key that creates the above new relationship suggestions. [27][28].

Approach 2 - Affiliation network

Here this approach visualizes the researcher Affiliation network and suggests the connectivity of the authors around that related affiliation. The input of this module is the author name and affiliation of each author.

4.5 Collaborators Recommendation Module

Finding best matching collaborators is not an easy task for the researchers. For the researchers who are new to the research field it is more difficult because of the lack of experience. To overcome this top matching collaborators are being recommended in this system for the ease of users. When recommending the top matching collaborators for the newly logged user, the grade of relevance is being considered as well as connectivity because good recommendation based on these two [17][23][21].

In the system profiling module provides all users in the system and when a new user logs in to the system the relatedness of each user to the newly logged one from the subject area is being generated by using Spearman's correlation method.

From each of those users a particular amount of the most related ones which bear a value greater than 7 are being separated and append those users into a new array named as (r). This is how the relevance is being measured according to the user profile.

To measure connectivity, the input is being given from the social network connectivity module. From connectivity module, 20 of the most relevant users from connectivity (c) is being sorted out.

By using FuzzyWuzzy string matching algorithm, the users in r and the users in c are being compared and the common set of users will be given with a fuzzy string matching value out of 100.
Among that common set array of names and values, the names which holds 100 of value will be appended to a dictionary because the particular user is in the name set which has been given according to the connectivity and relevance.

Finally a dictionary is being sorted in descending order considering their relevance value (r) and connectivity value (c) and the top 5 users will be recommended to the newly logged user which has the highest relevance and connectivity.

If the numbers of top matching set of users are less than 5, the lack amount of users will be selected from the remaining set of users which are not in the recommended list and the next highest scored users will be recommended as collaborators.

4.6 Summary

This chapter describes the abstract level architecture of the proposed system in terms of their input, process and output. Apart from that it provides summarizes how each module assists in achieving its objectives and the provide solution as a system.

Chapter 05

Analysis & Design

5.1 Introduction

Previous chapter provided a description about our approach of implementing the System. This chapter provides a description about the system analysis and the design.

5.2 Modules of the system

This system is going to be implemented in four modules

- ✓ Profiling
- ✓ Recommendation of journals and conference for manuscript
- ✓ Social Network Analysis
- ✓ Collaborators recommendation

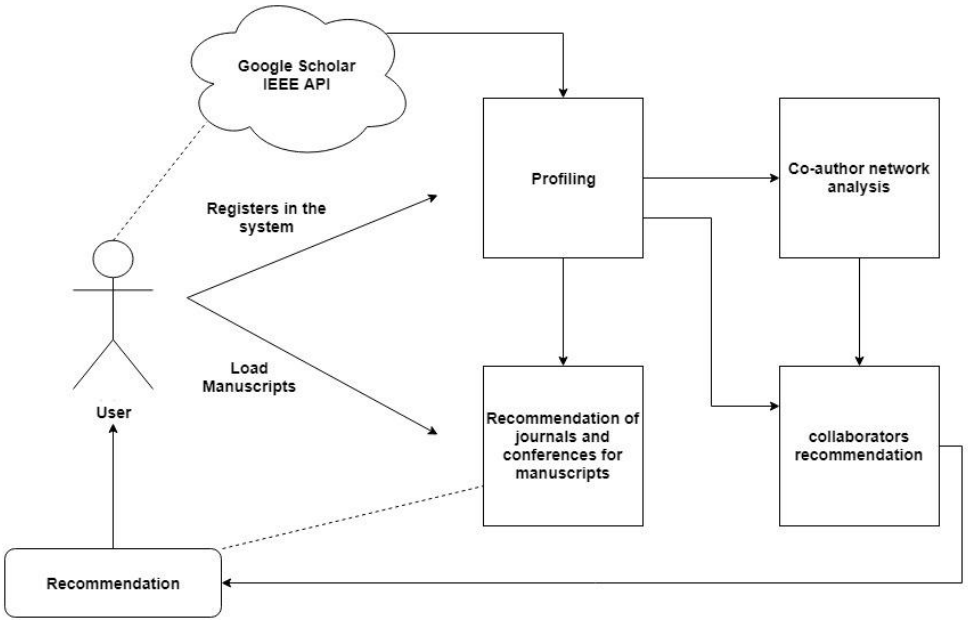


Figure 5.1: Intelligence Research Assistant

5.2.1 Profiling Module

Expert profiling is the process of determining key attributes that can be used to characterize a given expert. The objective of expert profiling is to extract expertise of researchers/experts [29]. Profiling module supplies data to the other three modules because the system is a profile-based recommendation system. In here the data will be extracted from the sources which contain the details of the researches as Google scholar and IEEE through API. Then the extracted data will be preprocessed and a most suitable profile for the user will be generated. Below figure include the high level diagram of the profiling module to show how it works in the Intelligent Research Assistant.

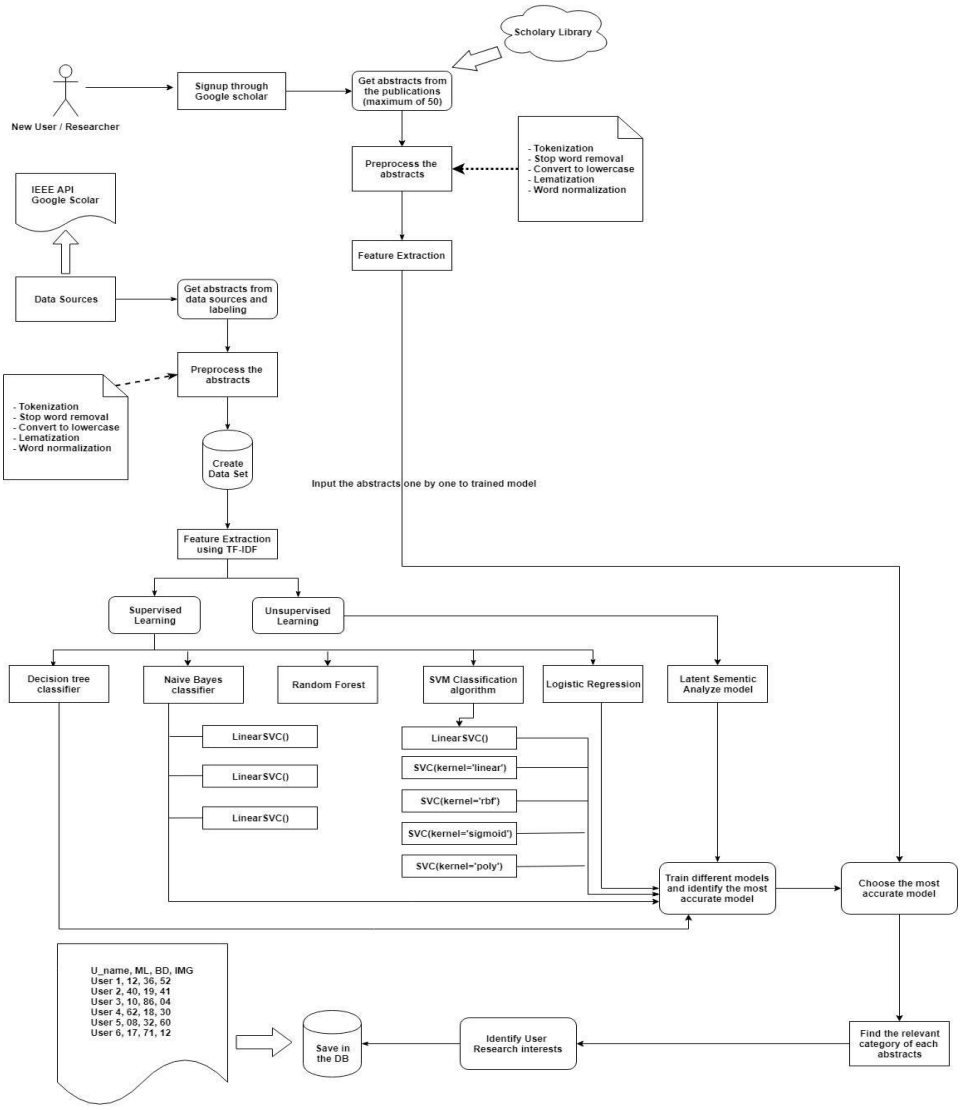


Figure 5.2 Profiling module structure

In this module, when a user signup to the system it gets the abstracts from the publications that published by the user from the Google Scholar through scholary library. After that preprocessing process will be started. Under the preprocessing it does tokenization, lemmatization, stop word removal. After that we get a quality data set and then the system start the features extraction process using TF-IDF.

After extract the features, system put the collected abstracts one by one into trained model. In here we used some of supervised learning classification algorithms and unsupervised Topic modeling approach name Latent Semantic Analysis (LSA). We chose the most accurate model from all those models and using that model, it decide the relevant category for each abstract. In here we identified 3 main research areas as the first level of classification as Machine Learning, Big Data, and Image Processing. After get the each abstract's relevant category, we calculate their final value using the occurrences for 3 different classes out of 100.

After that we go into some deep as the second level of categorization. In here we identified research areas as Machine Learning, Big Data, Data Mining, Computer Vision, Bioinformatics, and Artificial Intelligence.

5.2.2 Recommendation of journals for manuscript module

Recommendation of journals for manuscript module is provided quality of journals which matching with manuscript. The main aim of the productivity index in journal recommendation is to help researcher's find the most effective journals

The basic structure of the implementation is as follows,

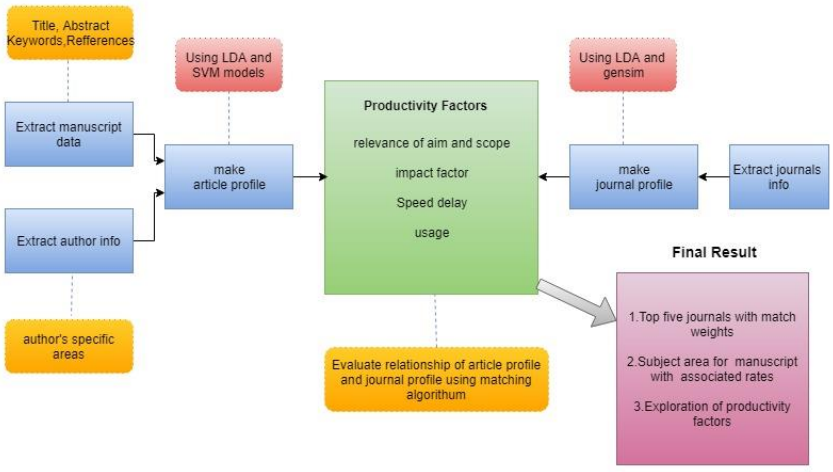


Figure 5.3: Diagram of the basic structure of the implementation

- **Data source**

Used springer open journals as a data source. It has two hundred more journals and different categories. Using web scrapping fetch the data from springer and train the modules. When is getting theme from manuscript and another dataset has to be trained module and it is included 14000 more data under the 3 categories.

- **Manuscript Profiling**

The objective of manuscript profiling is to extract relevant features including keywords and subject categories that can be used to represent manuscripts uniquely. Keywords are extracted from standard sections of a manuscript. It's mean extract a theme from title, keywords and abstract of paper using LDA to SVM models.it is a standard way.

- **Web scraping**

Using the generated quarries scraping get the search result from the digital library and go through each result item and find the highest probability of the similarity between the content of the manuscript and articles and they are considered as relevant articles to the researcher.

This crawler has the ability to identify search result items which are crawled before. Then it is reducing the processing time.

- **Journal Profiling**

The objective of journal profiling is to extract a journal's key attributes including key disciplines that it belongs to as well as keywords covered by the published journal. In this step included pre-processing part, Building dictionary, Feature extraction and LDA model training.

- **Pre-processing**

Data pre-processing part is used before generate manuscript profile and before generate journal profile. The purpose is to remove any unwanted words or characters which are written for human readability, but won't contribute to topic modeling in any way. There are mainly two steps that need to be done on word level. Those are removal of stop words and lemmatization. Stop words like "and, if, the" etc. These words are very common in English sentence therefore should be

removed from the articles. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. For example, “include”, “includes,” and “included” would all be represented as “include”.

- **Building word dictionary**

The vocabulary has to be built of the corpus and it has included all the unique word of the article. All words have IDs and it will be stored in corpus. Gensim library is used for build the dictionary. There are two additional steps:

```
(Topic 29, u'0.079*"die" + 0.060*"age"+ 0.059*"bear" + 0.029*"year" +
0.024*"death" +0.018*"january" + 0.018*"marry" + 0.016*"years" +0.016*"cancer"
+ 0.014*"february"')
(Topic 30,u'0.102*"south" + 0.050*"new" + 0.040*"west" +0.036*"north"
+ 0.034*"park" + 0.030*"east" +0.027*"wales" + 0.026*"division" +
0.020*"coast"+ 0.016*"australia"')
(Topic 39,u'0.041*"black" + 0.040*"white" + 0.034*"red" +0.029*"bird"
+ 0.027*"blue" + 0.024*"green" +0.020*"ship" + 0.019*"fly" + 0.019*"brown" +0.018*"wear"'
```

Figure5.4: dictionary

- **Feature extraction (Bag of words)**

This gives the knowledge that comparative reports will have word tallies like one another. At the end of the day, the more comparable the words in two archives, the more comparative the reports can be. This will be generate word count vector for the dictionary. In this step can see words with frequency counts.

- **LDA Model Training**

In this step explain about train phase of topic modeling. Latent Dirchlet allocation is used for this step. It is unsupervised model and it get top place among topic modeling algorithms. It will discover latent themes in document. This model make following assumptions,

1. Creating mixture of topics from articles.

2. Each topic is a generative model which generates words of the vocabulary with certain probabilities

	Word 1	Word 2	Word ...	Word N
Topic 1	$P(w_1\|t_1)$	$P(w_2\|t_1)$	...	$P(w_N\|t_1)$
Topic 2	$P(w_1\|t_2)$	$P(w_2\|t_2)$	...	$P(w_N\|t_1)$
Topic 3	$P(w_1\|t_3)$	$P(w_2\|t_3)$	...	$P(w_N\|t_1)$
...	...	...	...	...
Topic K	$P(w_1\|t_K)$	$P(w_2\|t_M)$	...	$P(w_N\|t_1)$

Figure 5.5: Topic-Word Distribution

```
(Topic 29, u'0.079*"die" + 0.060*"age"+ 0.059*"bear" + 0.029*"year" +
0.024*"death" +0.018*"january" + 0.018*"marry" + 0.016*"years" +0.016*"cancer"
+ 0.014*"february"')
(Topic 30,u'0.102*"south" + 0.050*"new" + 0.040*"west" +0.036*"north"
+ 0.034*"park" + 0.030*"east" +0.027*"wales" + 0.026*"division" +
0.020*"coast"+ 0.016*"australia"')
(Topic 39,u'0.041*"black" + 0.040*"white" + 0.034*"red" +0.029*"bird"
+ 0.027*"blue" + 0.024*"green" +0.020*"ship" + 0.019*"fly" + 0.019*"brown" +0.018*"wear"'
```

Figure 5.6: Topic-Word Distribution with Frequency

- **Theme extraction from manuscript (combination of LDA and SVM models)**

In this steps explain about theme extraction of given manuscript. LDA model is provided topics with their frequencies. LAD model has some limitations. It cannot label and will change topic every time. Therefore using hybrid module to minimize these issues. It is LDA to SVM (support vector machine).SVM is supervised algorithm model and it can train the datasets. Also it can be labeled and finally will release higher accuracy result.

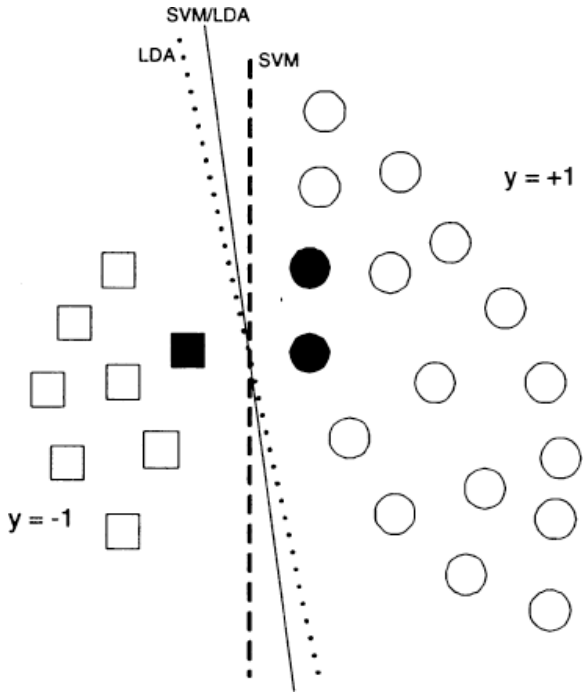


Figure 5.7: LDA to SVM model

- **Matching Algorithm**

In this step will be matched best solution for the problem. It means that in this step will be compare aim and scope relevance rate. Journal acceptance rate, usage and impact factor and finally will be recommend best journal for the manuscript. This part is represented productivity of the module.

5.2.3 The connectivity detection module

The social network analysis module is provided connectivity of authors which matching the author's affiliation and the co-authorship separately.

The basic structure of the implementation is as follows,

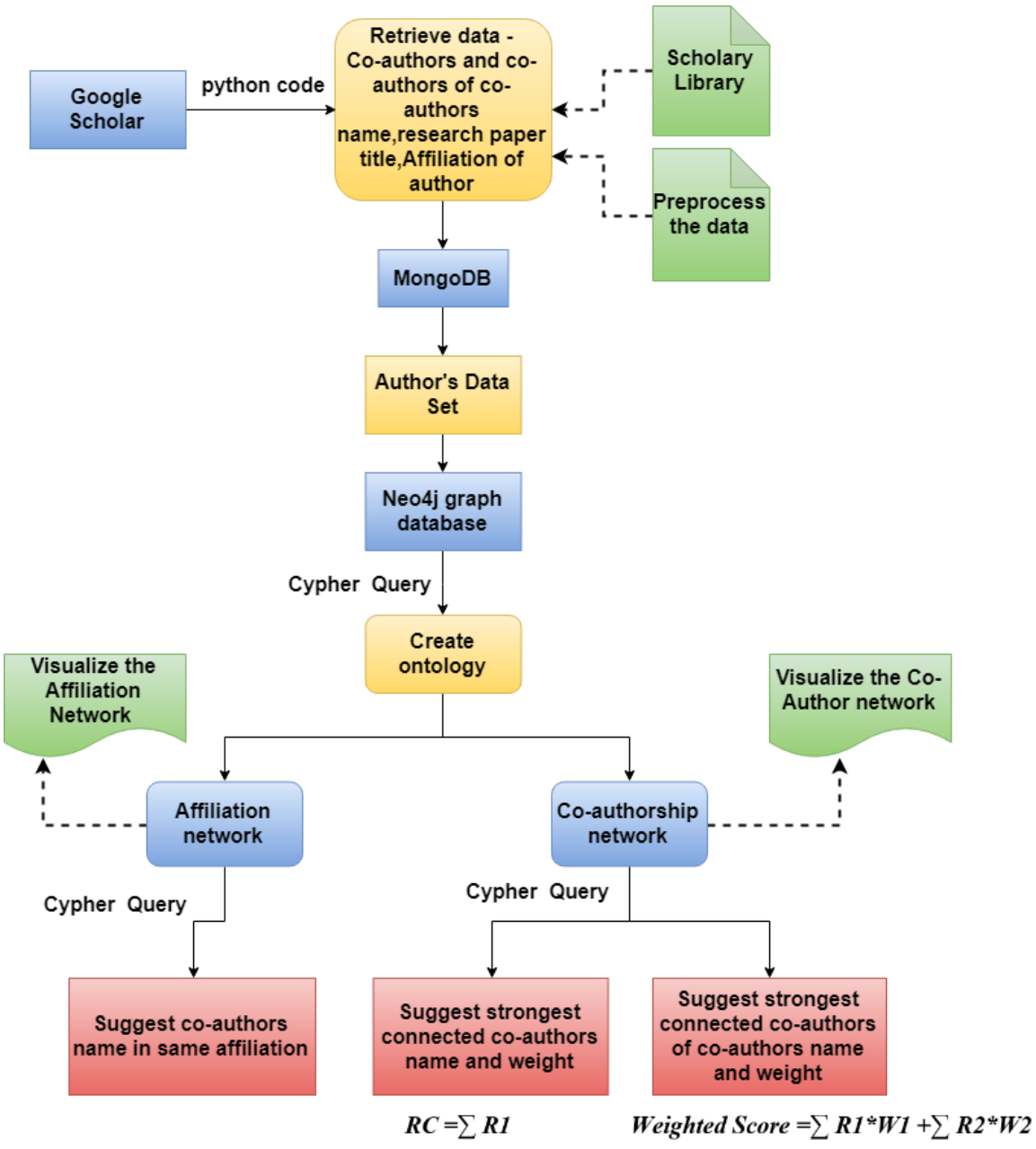


Figure 5.8: Diagram of the basic structure of the implementation

Data source → Use correct and detailed data is the key point in the network user's connectivity analysis. The module is tested on the data set which built from Google Scholar with the help of scholarly python library. Scholarly is a library that allows retrieving author and publication information from Google Scholar in a Pythonic way. The Author's affiliation, research paper title, co-authors and co-authors of co-authors names of each title are extracted to build data set with MongoDB.

Co-authorship data are available immediately after the publication of a paper. This fact allows for the development of large and relatively complete networks. However, the gathering of data is done using 10 research paper titles which have the highest citation from the author to gather data of the

Affiliation Data → Author's name and the affiliation are built relationship to generate the affiliation network.

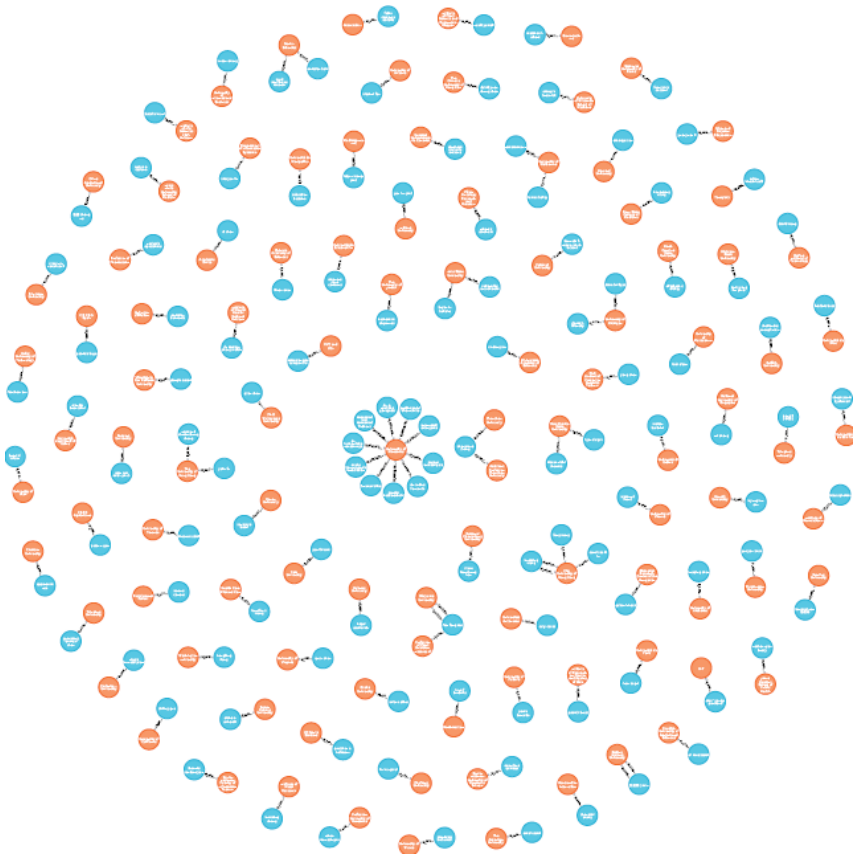


Figure 5.9: The affiliation Network graph containing 118 authors and 103 Affiliations

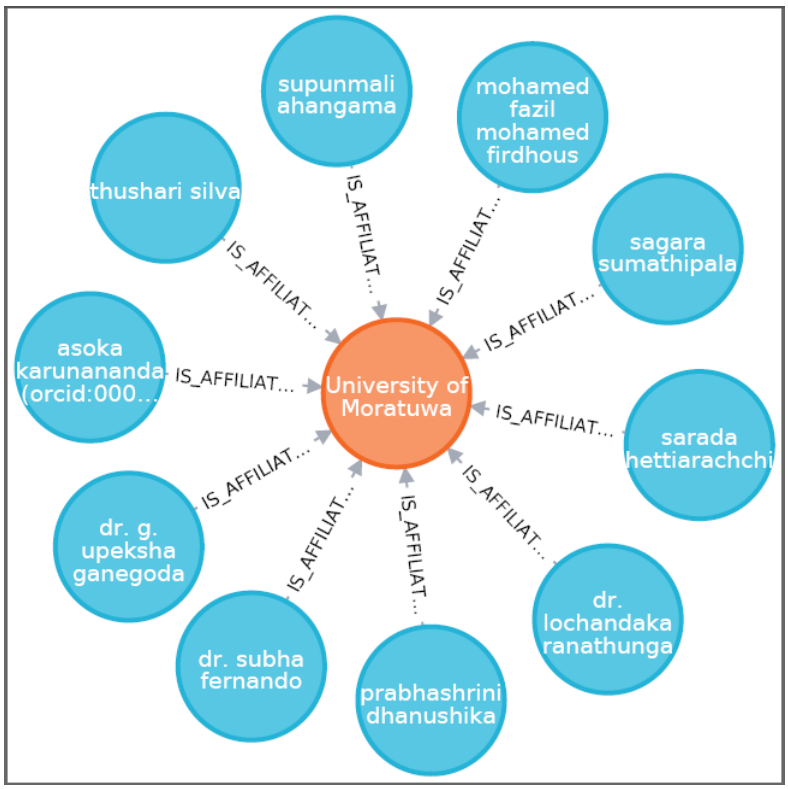


Figure 5.10: Section of the Affiliation network graph containing authors in blue and affiliation in orange

Above Graph shows that the example of how the affiliation network graph has visualized the connectivity of the author and the author's affiliation. There are ten authors who have a relationship with the University of Moratuwa. Identification of this connection helps to predict the collaborators to do future works.

Co-Authors data → Author and co-authors which related to each research paper title are built the Co-Authorship network. The relationship between the authors and the co-authors is built through the number of research paper title they have co-authored.

The analysis starts from calculating the strongest connected co-author for a particular researcher (user) and it extends to find out the co-authors of co-authors which has the strongest connection.

Here, study the problem of predicting future collaborators through the existing authors in the co-authorship network.

In co-authorship network represent the relationship between people by using a graph comprised of nodes and edges. Each node represents authors and co-authors and each edge that in or out of nodes represent the relationship. Here node represents authors and co-authors and edges are represent the research paper title.

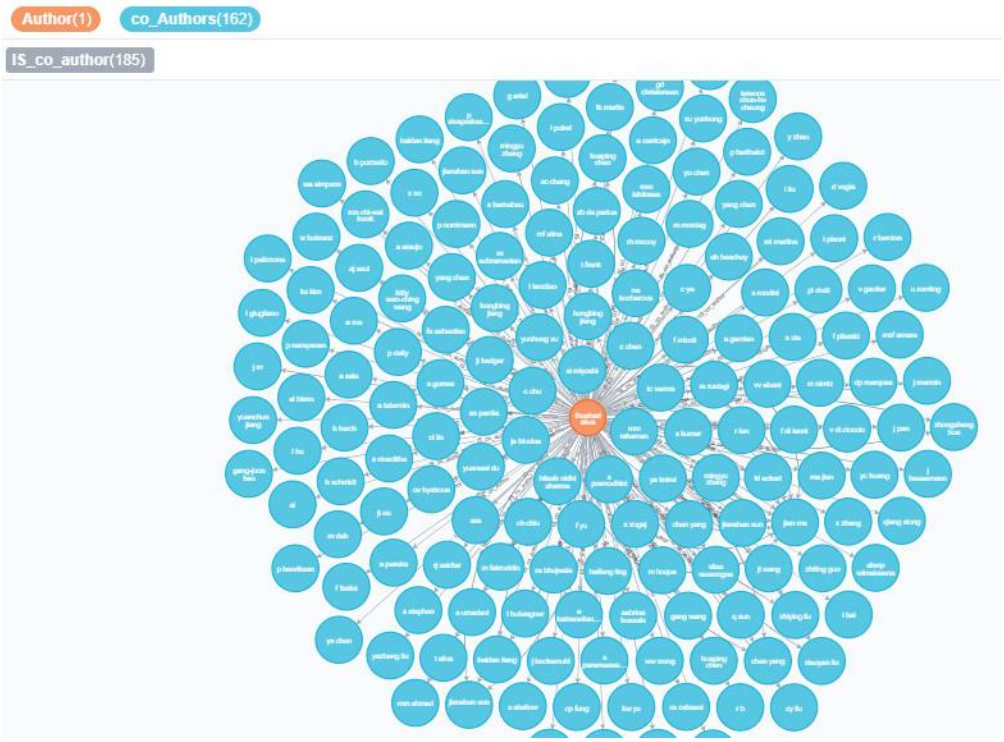


Figure 5.11: Graph containing one Author (user) and co-authors

Above graph shows that one of the author's relationships with co-authors. It contains one author and 162 co-authors with 185 relationships. The author is represented in orange and co-authors are blue.

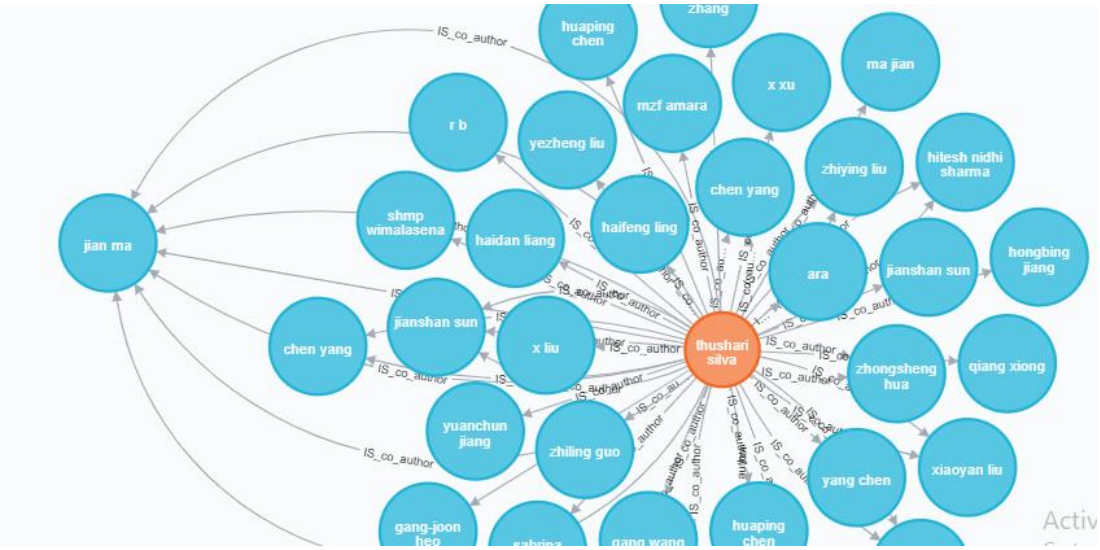


Figure 5.12: sample graph containing one Author and co-authors with relationships

Above graph contains one author and 31 co-authors with 68 relationships. The author is represented in orange and co-authors are blue. The relationship named "IS_CO-AUTHOR OF" in the graph corresponds to the research paper title. As an example above graph shows co-author named Jian ma have co-authored with an author named Thushari Silva. Also, it shows seven relationships between them. It means Jian ma have co-authored with seven research papers as well as shows how strong the relationship they have than other co-authors.

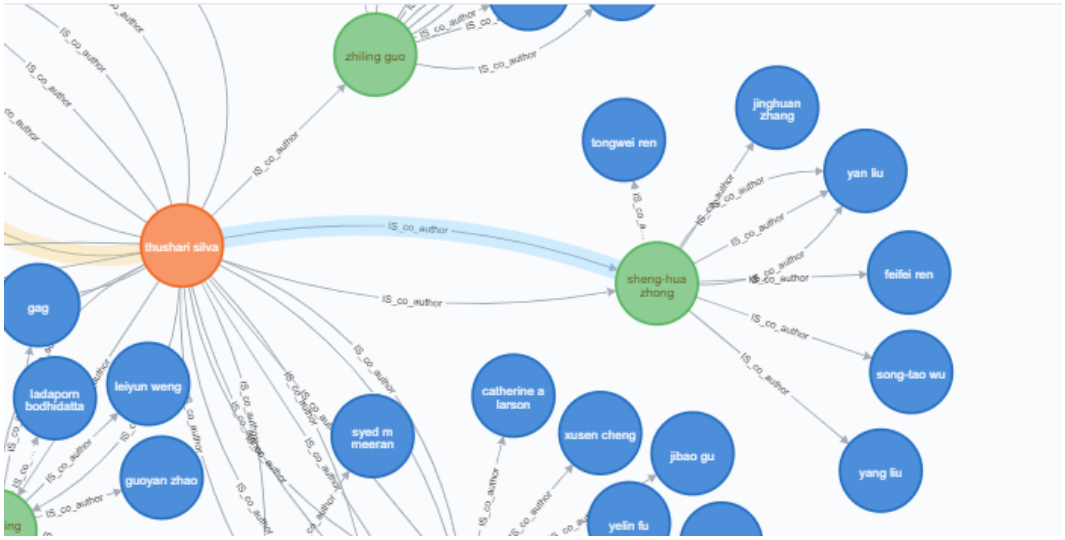


Figure 5.13: section of co-authorship graph containing co-authors of co-authors relationship

Using co-authorship network able to predict co-authors of co-authors relationship not only coauthors.

5.2.4 Collaborators Recommendation Module

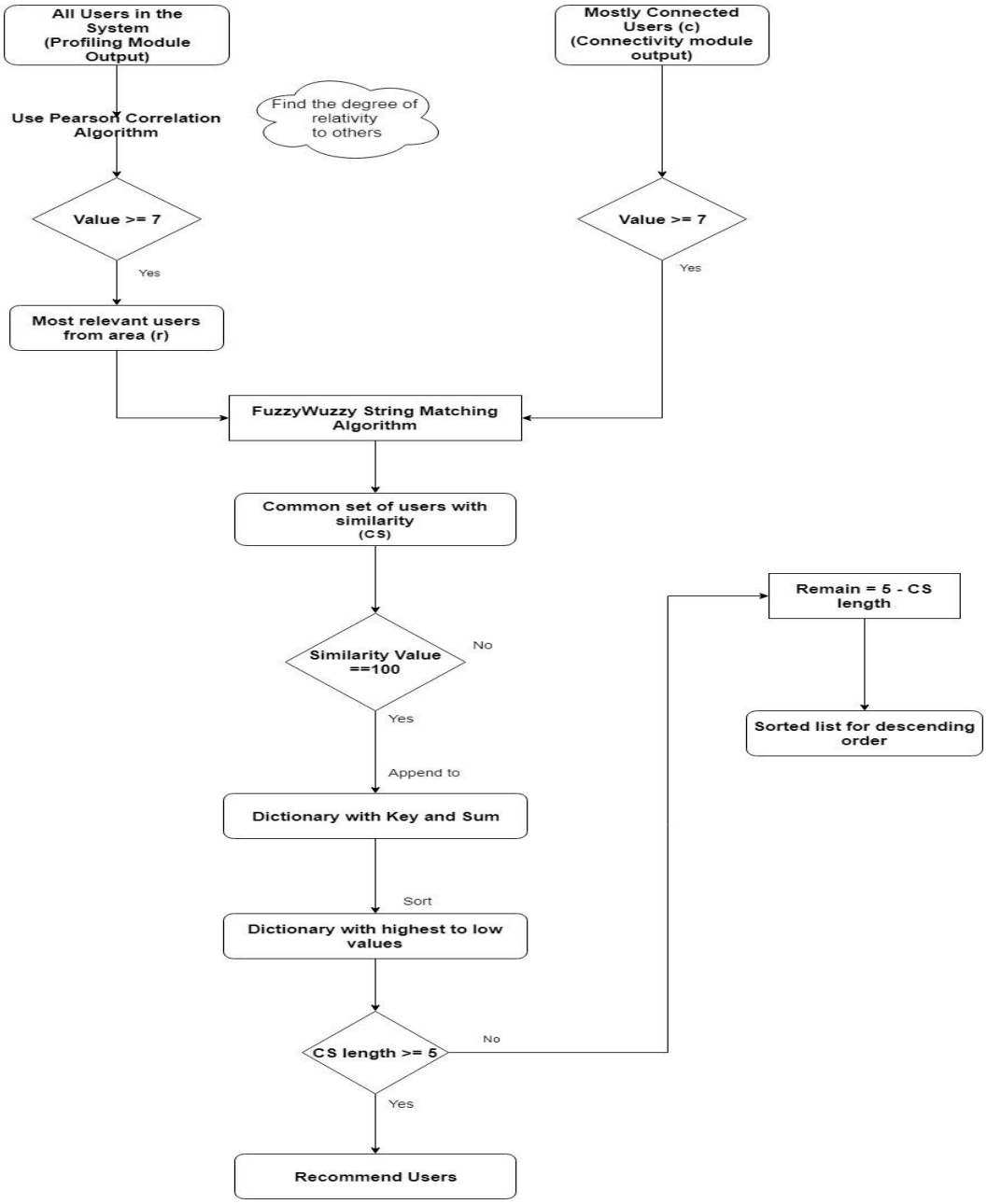


Figure 5.14: Collaborators recommendation module diagram (Pearson's correlation)

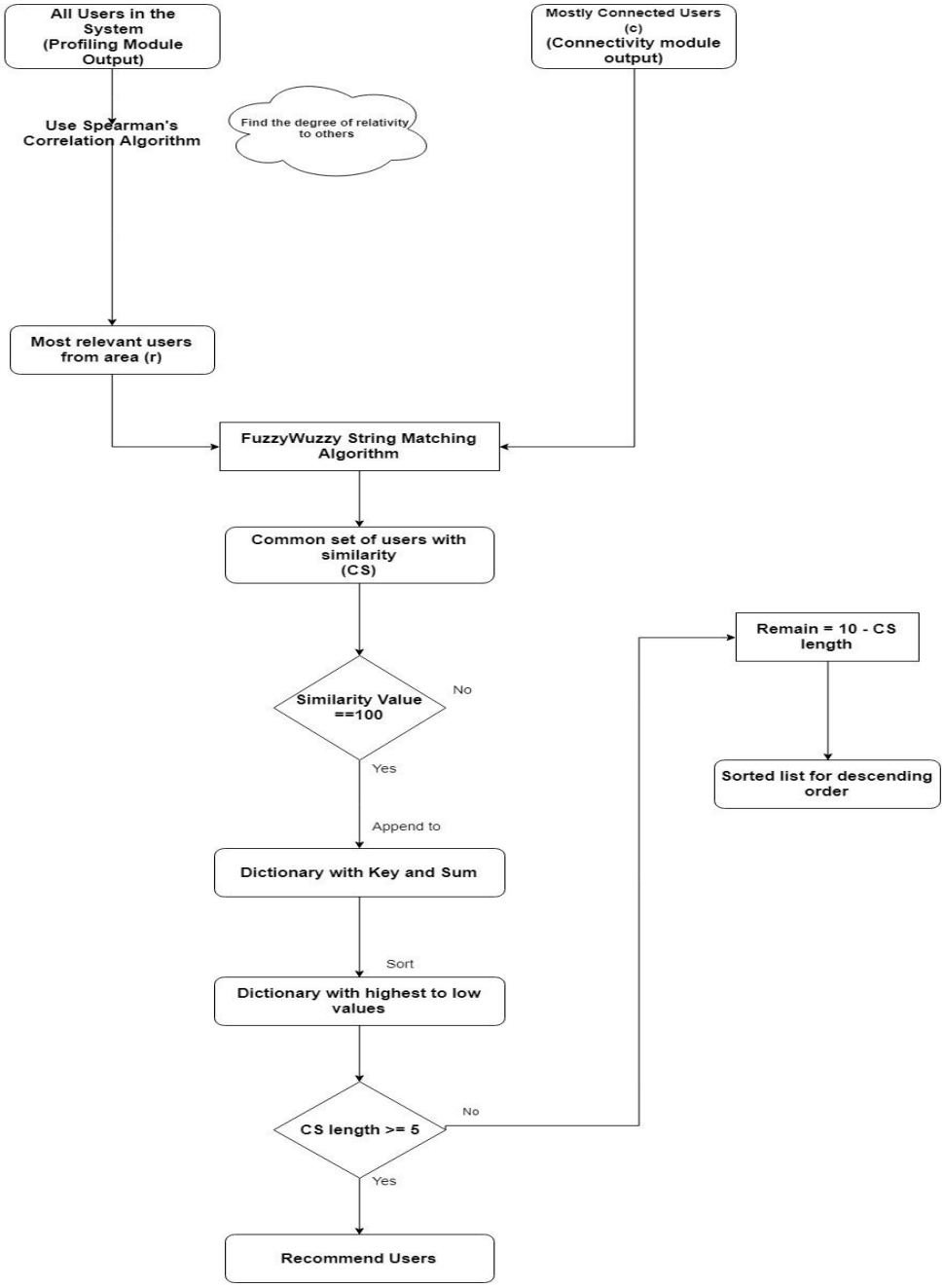


Figure 5.15: Collaborators recommendation module diagram (Spearman's correlation)

When the new user is logged into the system, the degree of his relativity for each subject module will be collected by the profile module. After that the user will be compared with each user in the system and by using the correlation algorithm the users who have the most similarity out of them will be collected. Among those users, the ones who have correlation value more than 7 will be selected if the Pearson's correlation algorithm. And is there are no users as the user requested to suggest as

collaborators then the relativity value (r) will be remain same and Connectivity value (c) will be decreased one by one. If Spearman's algorithm is used the top users according to the situation will be selected. Now, most relevant users from the area (r) are already selected. Then the most connected users (c) will be given by the Social connections module.

Now the common set of users for both r and c should be found. Fuzzy string matching algorithm is used here for that purpose. By matching each user of two data sets with each other the users who have prefect match which is 100% will be found by using the IDs. After that the common users are being appended to a dictionary and the 5 users will be given to the user from the system. The user can change the number of collaborators should be suggested to them. I there are no users to recommend, from the sorted CS list, the excess will be recommended.

5.3 Summary

The chapter five is described the analysis and the design of all four modules with visualizations as well as the necessary steps with diagrams. It is provided comprehensive understanding about how the system process is carried out under different modules.

Chapter 06

Implementation

6.1 Introduction

This chapter describes the implementation of proposed solution under four modules that are profiling module, social network analysis module, collaborators recommendation module. The details on how the previously mentioned technologies are used to implement the identified solution are included here. All the significant and important implementation steps and algorithms are shown with comprehensive details.

6.2 Profiling Module

6.2.1 Get user abstracts

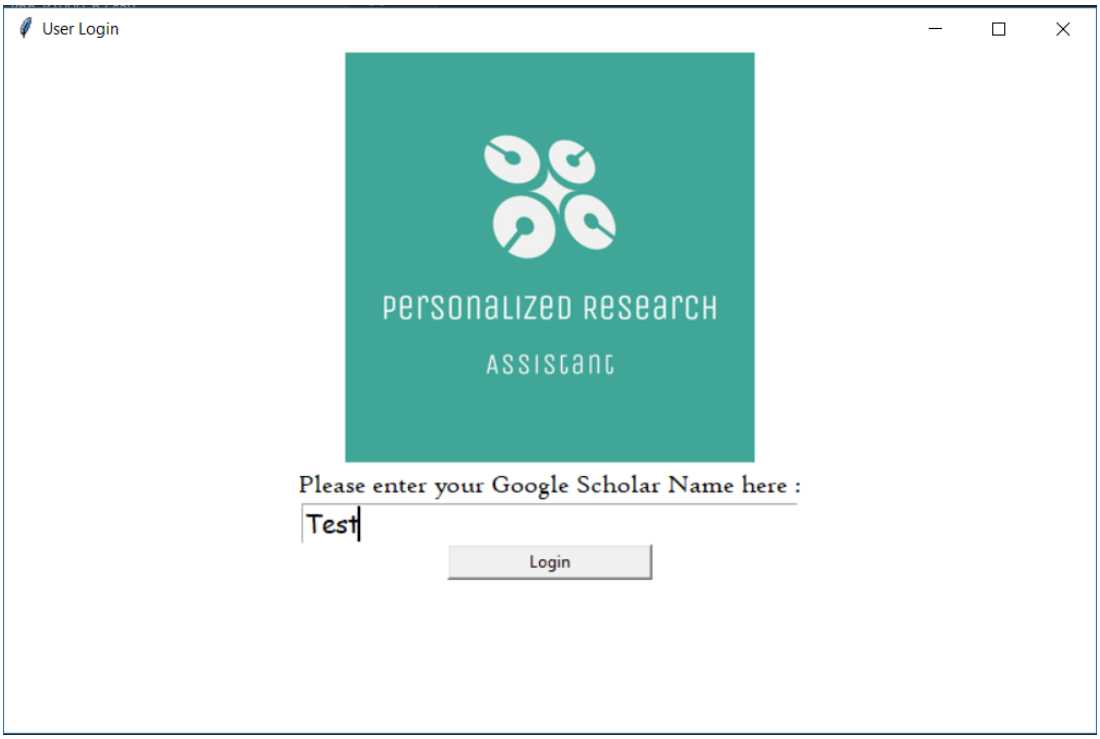


Figure 6.1 - User Interface to get Google Scholar name of the user

When a user login to the system, system get the Google Scholar name and get the user abstracts up to 100 through following code.

```
import csv
import scholarly

def get_user_details(u_name):

    user_details = []
    last_id = ''
    with open('datasets/users.csv', encoding="utf-8") as users data:
        users = csv.reader(users data, delimiter=',')
        for row in users:
            last_id = row[0]

    u_id = int(last_id) + 1
    search_query = scholarly.search_author(u_name)
    author = next(search_query).fill()

    user_affliation = author.affiliation
    user_key_words = author.interests

    sentence = ''
    for word in user_key_words:
        sentence = sentence + word + ' '

    x = author.publications.__len__()

    if (x < 100):
        num_pub = x
    else:
        num_pub = 100

    abstracts = []
    for count in range(0, num_pub):
        pub = author.publications[count].fill()
        abst = pub.bib.get('abstract')
        abst = str(abst)

        abstracts.append(abst)

    user_details.append(u_id)
    user_details.append(u_name)
    user_details.append(user_affliation)

    print(user_details)
    preprocess_abstracts(abstracts)
```

Figure 6.2: Implementation of get the user abstracts from publications

6.2.2 Preprocess abstracts

After getting the abstracts from Google scholar, those collected abstracts send to preprocess functionality.

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
from nltk.stem import  PorterStemmer
stemmer = PorterStemmer()

def preprocess_abstracts(abstracts):

    bad_chars = ['<div class="gsc_vcd_value"', ' id="gsc_vcd_descr">', '<div
class="gsh_small">', '.', '”', '‘', '^', '<div class="gsh_csp">',
                 'gsh_cspExpert', 'gsh_csp', '</div>', '/', ':', '-', '(', ')', '0',
'1', '2', '3', '4', '5', '6', '7', '8', '9', ';', '[', ']', ',',
                 '"', '<', '>', '"', '=', '…', 'div', "’", '\'']

    bad_words = ['we', 'the', 'in', 'a', 'during', 'and', 'an', 'this', 'these',
'that', 'paper', 'review', 'more', 'our', 'purpose', 'from', 'to',
```

```
                'ass', 'as', 'public', 'live', 'study', 'along', 'purpose', 'open',
'job', 'main', 'h', 'couple', 'year', 'service', 'user', 'new',
                'present', 'report', 'use', 'within', 'exist', 'strong', 'using',
'work', 'task', 'always']

    preprocessed_abs = []
    for abstract in abstracts:
        for i in bad chars:
            abstract = abstract.replace(i, '')

        tokens = nltk.word_tokenize(abstract)

        lower_words = [lw_word.lower() for lw_word in tokens]
        remove bad words = [word for word in lower words if not word in bad words]

        stop_words = set(stopwords.words('english'))
        with_out_stop_words = [w for w in remove_bad_words if not w in stop_words]

        # stem abs = [stemmer.stem(word) for word in with out stop words]

        final abs = ''
        for word in with_out_stop_words:
            final_abs = final_abs + lemmatizer.lemmatize(word) + ' '
        preprocessed_abs.append(final_abs)
```

Figure 6.3: Implementation of preprocessing data

Sample of preprocessed abstract as follows.

```
Id, abstract, label
ML 1,semisupervised learning learning paradigm concerned computer natural system human learn
presence labeled unlabeled data traditionally learning studied either unsupervised paradigm eg
clustering outlier detection data unlabeled supervised paradigm eg classification regression data
labeled goal semisupervised learning understand combining labeled unlabeled data may change
learning behavior design algorithm take advantage combination semisupervised learning great
interest machine learning data mining readily available unlabeled data improve supervised learning
task labeled data scarce expensive semisupervised learning also show potential quantitative tool
understand human category learning input selfevidently unlabeled introductory book popular
semisupervised learning model including selftraining mixture model cotraining multiview learning
graphbased method semisupervised support vector machine model discus basic mathematical
formulation success semisupervised learning depends critically underlying assumption emphasize
assumption made model give counterexample appropriate demonstrate limitation different model
addition discus semisupervised learning cognitive psychology finally give computational learning
theoretic perspective semisupervised learning conclude book brief discussion question field table
content introduction statistical machine learning overview semisupervised learning mixture model
em cotraining graphbased semisupervised learning semisupervised support vector machine human
semisupervised learning theory outlook ,1
ML_2,learning rank refers machine learning technique training model ranking learning rank useful
many application information retrieval natural language processing data mining intensive study
conducted problem recently significant progress made lecture give introduction area including
fundamental problem major approach theory application future author begin showing various ranking
problem information retrieval natural language processing formalized two basic ranking task namely
ranking creation simply ranking ranking aggregation ranking creation given request one want
generate ranking list offering based feature derived request offering ranking aggregation given
request well number ranking list offering one want generate ranking list offering ranking creation
ranking major problem learning rank usually formalized supervised learning author give detailed
explanation learning ranking creation ranking aggregation including training testing evaluation
feature creation major approach many method proposed ranking creation method categorized pointwise
pairwise listwise approach according loss function employ also categorized according technique
employ svm based boosting based neural network based approach author also introduces popular
learning rank method detail include prank oc svm mcrank ranking svm ir svm gbrank ranknet listnet
& amp listmle adarank svm map softrank lambdarank lambdamart borda count markov chain cranking
author explains several example application learning rank including web search collaborative
filtering definition search keyphrase extraction query dependent summarization reranking machine
translation formulation learning ranking creation given statistical learning framework ongoing
future research direction learning rank also discussed table content learning rank learning
ranking creation learning ranking aggregation method learning rank application learning rank
theory learning rank ongoing future ,1
ML 3,nonconvex optimization machine learning take indepth look basic nonconvex optimization
application machine learning introduces rich literature area well equips reader tool technique
needed apply analyze simple powerful procedure nonconvex problem nonconvex optimization machine
learning selfcontained possible losing focus topic nonconvex optimization technique monograph
initiate discussion entire chapter devoted presenting tutoriallike treatment basic concept convex
analysis optimization well nonconvex counterpart monograph concludes look four interesting
application area machine learning signal processing exploring nonconvex optimization technique
introduced earlier used solve problem monograph also contains topic discussed exercise figure
designed engage reader well extensive bibliographic note pointing towards classical work recent
advance nonconvex optimization machine learning used semesterlength course basic nonconvex
```

```
optimization application machine learning hand also possible cherry pick inidual portion chapter
sparse recovery em algorithm inclusion broader course several course machine learning optimization
signal processing may benefit inclusion topic ,1
```

Figure 6.4: Sample of preprocessed abstract

6.2.3 Train and test the dataset

- **Dataset**

As first step, 150 user names are being collected from the google scholar manually for all predefined 3 classes as 50 per each. Then those names are being looped one by one to script and it collected 25 abstracts from each user. When it is saved to the given csv file the abstract is also being labeled based on that user's interesting area according to the predefined classes. But some of the abstracts include the quality problems as follows. There are lots of unwanted sentences created a result of some character problems.

Ex: Some of texts in the 1 machine learning abstract

″<div class=""gsc_vcd_value"" id=""gsc_vcd_descr""><div class=""gsh_small""> <div class=""gsh_csp"">A search for the Standard Model Higgs boson in proton–proton collisions with the ATLAS detector at the LHC is presented. The datasets used correspond to integrated luminosities of approximately 4.8 fb^{−1} collected at <svg aria-label=""s=7 TeV"" class=""gs_fsvg"" height=""15px"" style=""vertical-align:-3px;"" width=""70px""><g transform=""matrix(0.01400, 0.00000, 0.00000, 0.01400, 0.00000, 11.16499)""><g><g><path d=""M 719 -1954 L 311 -1057 L 190 -1149 Q 184 -1155 176 -1155 Q 167 -1155 157 -1146 T 147 -1126 Q 147 -1116 156 -1110 L 399 -926 Q 405 -920 414 -920 Q 427 -920 434 -934 L 801 -1741 L 1671 63 Q 1681 82 1706 82 Q 1723 82 1735 70 T 1747 41 Q 1747 31 1745 27 L 793 -1948 Q 780 -1966 760 -1966 H 737 Q 727 -1966 719 -1954 Z "" transform=""matrix(0.48828, 0.00000, 0.00000, -0.48828, 0.00000, -717.49945)""></path><g transform=""translate(833.33002, 0.00000)""><path d=""M 178 125 Q 233 31 399 31 Q 471 31 536 55 T 643 129 T 686 248 Q 686 301 648 335 T 555 381 L 444 403 Q 368 422 319 474 T 270 600 Q 270 691 319 761 T 450 868 T 618 905 Q 711 905 784 860 T 858 729 Q 858 682 831 646 T 758 610 Q 731 610 711 627 T 692 672 Q 692 696 705 718 T 741 754 T 788 768 Q 770 812 720 832 T 614 852 Q 562 852 510 831 T 426 769 T 395 674 Q 395 637 421 609 T 485 569 L 604 545 Q 661 533 708 502 T 783 425 T 811 319 Q 811 243 768 169 T 664 51 Q 555 -23 397 -23 Q 288 -23 197 27 T 106 176 Q 106 232 138 273 T 227 315 Q 260 315 282 295 T 305 242 Q 305 195 270 160 T 188 125 H 178 Z "" transform=""scale(0.48828, -0.48828)""></path><line style=""stroke-width:40.00000"" transform=""translate(0.00000, -717.49945)"" x1=""0"" x2=""468.78000"" y1=""0"" y2=""0""></line></g></g><g transform=""translate(1302.10999, 0.00000)""><path d=""M 154 272 Q 137 272 126 285 T 115 313 Q 115 330 126 342 T 154 354 H 1440 Q 1455 354 1466 342 T 1477 313 Q 1477 298 1466 285 T 1440 272 H 154 Z M 154 670 Q 137 670 126 682 T 115 711 Q 115 726 126 739 T 154 752 H 1440 Q 1455 752 1466 739 T 1477 711 Q 1477 694 1466 682 T 1440 670 H 154 Z "" transform=""matrix(0.48828, 0.00000, 0.00000, -0.48828, 277.77780, 0.00000)""></path>in 2012. Individual searches in the channels<path d=""M 604 -401 Q 604 -355 624 -267 T 674 -80 T 725 84 Q 733 186 733 238 Q 733 370 707 483 T 609 673 T 408 750 Q 340 750 272 721 T 156 640 T 90 522 Q 86 510 74 510 H 49 Q 33 510 33 528 V 535 Q 56 627 114 712 T 257 851 T 436 905 Q 532 905 602 840 T 714 675 T

774 473 T 793 281 Q 896 561 1024 811 L 1055 877 Q 1061 883 1069 883 H 1094 Q 1110 883 1110 864 Q 1110 860 1106 852 L 1075 788 Q 986 611 911 429 T 784 78 Q 776 9 753 -111 T 699 -335 T 635 -442 Q 604 -442 604 -401 Z "" transform=""matrix(0.48828, 0.00000, 0.00000, -0.48828, 3041.35864, 0.00000)""></path></g></g></svg> and H→WW(∗)→evμv in the 8 TeV data are combined with previously published results of searches.

But all of the abstracts are not in this way. There are about 30% of the abstracts included that kind of unnecessary words. After that this dataset is being processed and some models are being trained using preprocessed dataset. Those techniques will discuss in the next sub topic.

As some quality problems and accuracy problems of the dataset collected from the Google scholar, the abstracts are being collected again from the IEEE using their API. But they only provide maximum of 200 records for a query. So we collected are being collected for one category by changing some parameters of the requests.

Ex: Those are the 4 requests to get 800 machine learning abstracts from the IEEE API with different parameters.

```
Url_ml_one =
'http://ieeexploreapi.ieee.org/api/v1/search/articles?apikey=zkn22d5qwu42d44tkmmn4sch&format=json&max_records=200&start_record=1&sort_order=asc&sort_field=author&abstract=Machine+learning'
url_ml_two =
'http://ieeexploreapi.ieee.org/api/v1/search/articles?apikey=zkn22d5qwu42d44tkmmn4sch&format=json&max_records=200&start_record=1&sort_order=asc&sort_field=article_title&abstract=Machine+learning'
url_ml_three =
'http://ieeexploreapi.ieee.org/api/v1/search/articles?apikey=zkn22d5qwu42d44tkmmn4sch&format=json&max_records=200&start_record=1&sort_order=asc&sort_field=publication_title&abstract=Machine+learning'
url_ml_four =
'http://ieeexploreapi.ieee.org/api/v1/search/articles?apikey=zkn22d5qwu42d44tkmmn4sch&format=json&max_records=200&start_record=1&sort_order=desc&sort_field=article_number&abstract=Machine+learning'
```

One of machine learning abstract got through the API as follow.

"Semi-supervised learning is a learning paradigm concerned with the study of how computers and natural systems such as humans learn in the presence of both labeled and unlabeled data. Traditionally, learning has been studied either in the unsupervised paradigm (e.g., clustering, outlier detection) where all the data are unlabeled, or in the supervised paradigm (e.g., classification, regression) where all the data are labeled. The goal of semi-supervised learning is to understand how combining labeled and unlabeled data may change the learning behavior, and design algorithms that take advantage of such a combination. Semi-supervised learning is of great interest in machine learning and data mining because it can use readily available unlabeled data to improve supervised learning tasks when the labeled data are scarce or expensive. Semi-supervised learning also shows potential as a quantitative tool to understand human category learning, where most of the input is

self-evidently unlabeled. In this introductory book, we present some popular semi-supervised learning models, including self-training, mixture models, co-training and multiview learning, graph-based methods, and semi-supervised support vector machines. For each model, we discuss its basic mathematical formulation. The success of semi-supervised learning depends critically on some underlying assumptions. We emphasize the assumptions made by each model and give counterexamples when appropriate to demonstrate the limitations of the different models. In addition, we discuss semi-supervised learning for cognitive psychology. Finally, we give a computational learning theoretic perspective on semi-supervised learning, and we conclude the book with a brief discussion of open questions in the field. Table of Contents: Introduction to Statistical Machine Learning / Overview of Semi-Supervised Learning / Mixture Models and EM / Co-Training / Graph-Based Semi-Supervised Learning / Semi-Supervised Support Vector Machines / Human Semi-Supervised Learning / Theory and Outlook"

So the qualities of the abstracts are higher in the IEEE than the Google Scholar.

- **Preprocess the dataset**

When the abstracts are being collected through the automated scripts, the basic preprocessing method is being done to get a quality dataset to do most accurate predictions. [29] In here all the unnecessary words are being removed, punctuations, numbers etc. Use NLTK to preprocess the dataset. As well as own set of stop words are being used as well.

Predefined stopwords set

bad_chars = ['.', '”', '“', '^', '/', ':', '-', '(', ')', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ';', '[', ']', ',', '"', '<', '>', "", '=', '…', 'div', "'", '\"]

bad_words = ['we', 'the', 'in', 'a', 'during', 'and', 'an', 'this', 'these', 'that', 'paper', 'review', 'more', 'our', 'purpose', 'from', 'to', 'ass', 'as', 'public', 'live', 'study', 'along', 'purpose', 'open', 'job', 'main', 'h', 'couple', 'year', 'service', 'user', 'new', 'present', 'report', 'use', 'within', 'using', 'work', 'task', 'always']

From NLTK, tokenization is used to divide the abstracts into words. And also use the stop word from NLTK. Corpus and the own stop words set to remove unnecessary words in the abstracts. After that all the letters are being converted into lowercase using lower () function in python. After doing all these the words are being converted into its common root called as lemmatizing. This is very useful because it is very important to identify word frequencies in the dataset.

Examples for lemmatization.

studied = study

studying = study

Feature Extraction to get the correct decisions

- **Training the dataset**

For training the created different classification techniques are used. SVC with 4 different kernels such as linear, rbf, sigmoid, poly and LinearSVC are used under the support vector machines. Naïve bayes techniques like MultinomialNB, BernouliNB and GaussianNB also used. Apart from that Random forest, Logistic regression and Decision trees also divided. To train dataset using these different classifiers, first the dataset is being divided into 2 parts as 'abstract' and 'label'. After that each abstracts is being labeled when it preprocess to relevant category. After training the dataset with those different classifiers with changing different parameters, model for each classifier is saved.

Support Vector Machine Classifiers (Multi Class Classification)

- SVC with linear kernel

```
from sklearn.svm import SVC

X train, X test, y train, y test = train test split(data.abstract, data.lable,
test size=0.2)
pipeline = Pipeline([('vect', TfidfVectorizer(ngram range=(1, 1),
sublinear_tf=True)),
                     ('clf', SVC(kernel='linear', C=1))])

model = pipeline.fit(X_train, y_train)
```

- SVC with rbf kernel

```
from sklearn.svm import SVC

X train, X test, y train, y test = train test split(data.abstract, data.lable,
test size=0.2)
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('clf', SVC(kernel='rbf', gamma=0.7, C=1))])

model = pipeline.fit(X_train, y_train)
```

- SVC with sigmoid kernel

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', SVC(kernel="sigmoid", gamma=7,
probability=True))])
```

- SVC with poly kernel

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', SVC(kernel='poly', degree=6, C=1))])
```

- LinearSVC

```
from sklearn.svm import LinearSVC

X train, X test, y train, y test = train test split(data.abstract, data.lable, test size=0.2)
pipeline = Pipeline([('vect', TfidfVectorizer(ngram range=(1, 1), sublinear tf=True)),
                     ('clf', LinearSVC(C=1))])

model = pipeline.fit(X_train, y_train)
```

Naïve Bayes classifiers

- MultinomialNB

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', MultinomialNB())])
```

- GaussianNB

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', GaussianNB())])
```

- BernouliNB

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', BernoulliNB(alpha=7.0, binarize=0.0,
class_prior=None, fit_prior=True))])
```

Logistic regression classifier

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', LogisticRegression(random_state=0))])
```

Random forest classifier

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf',RandomForestClassifier(n_estimators=100,
max_depth=3, random_state=0))])
```

Decision tree classification

```
X_train, X_test, y_train, y_test = train_test_split(data.abstract,
data.lable, test_size=0.2)
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('clf', DecisionTreeClassifier(max_depth=2))])

model = pipeline.fit(X_train, y_train)
```

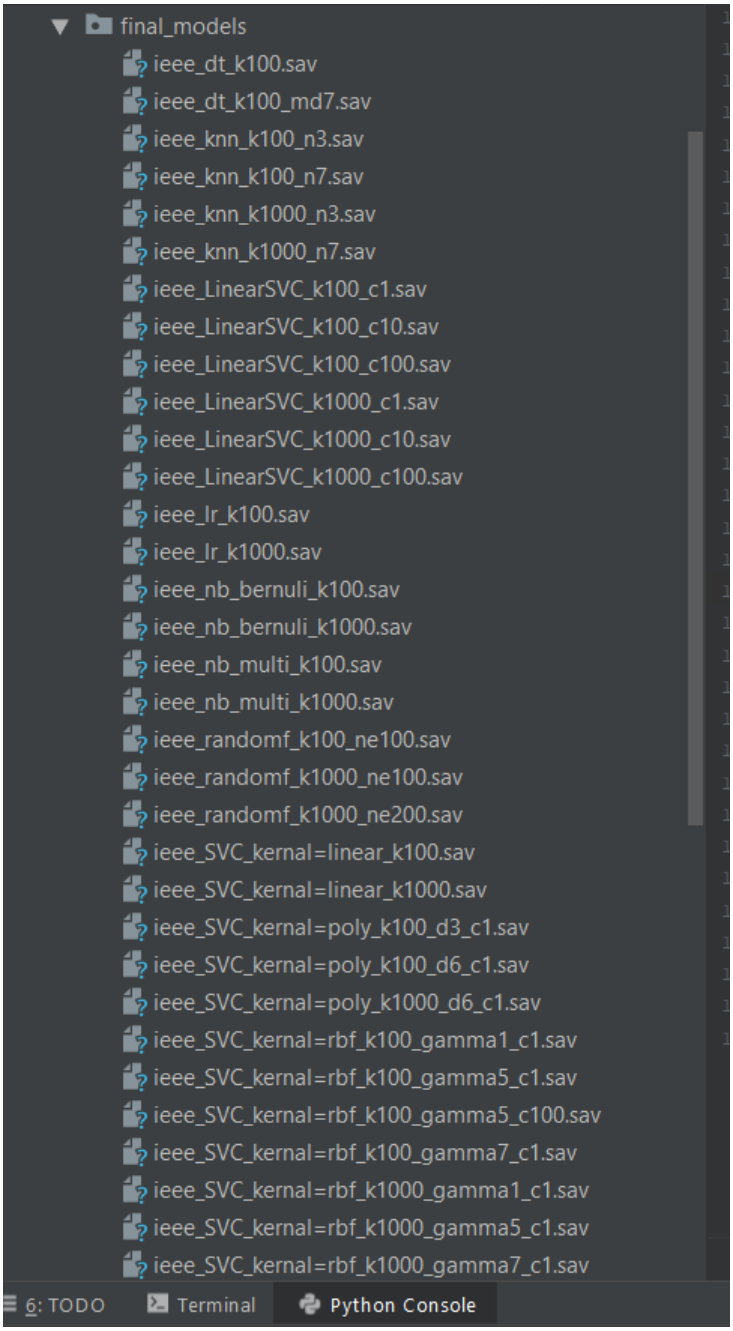


Figure 6.5 – Some set of models trained and saved

6.2.4 Find User's interest for defined research areas.

6.2.4.1 Based on total abstracts (Method 1)

First method of this project is predicting the abstract one by one to relevant classes. Then each category is divided from total abstracts and calculates the user interest as a percentage. Implementation of that method is as follows. (Results of this method is included in Appendix B)

```
def predict_interests(abstracts):

    model_file = 'model/ieee_SVC_kernal=sigmoid_prob.sav'
    loaded_model = pickle.load(open(model_file, 'rb'))


    ml = []
    bd = []
    dm = []
    cv = []
    bioin = []
    ai = []

    for abstract in abstracts:
        result = loaded_model.predict([abstract])
        if (result == 1):
            ml.append(result)
        if (result == 2):
            bd.append(result)
        if (result == 3):
            dm.append(result)
        if (result == 4):
            cv.append(result)
        if (result == 5):
            bioin.append(result)
        if (result == 6):
            ai.append(result)
        print(result)

    ml_len = ml.__len__()
    bd_len = bd.__len__()
    dm_len = dm.__len__()
    cv_len = cv.__len__()
    bio_len = bioin.__len__()
    ai_len = ai.__len__()

    denominator = abstracts.__len__()
    print(denominator)

    ml_score = round(((ml_len / denominator) * 100), 2)
    bd_score = round(((bd_len / denominator) * 100), 2)
    dm_score = round(((dm_len / denominator) * 100), 2)
    cv_score = round(((cv_len / denominator) * 100), 2)
    bio_score = round(((bio_len / denominator) * 100), 2)
    ai_score = round(((ai_len / denominator) * 100), 2)

    print(str(ml_score) + ', ' + str(bd_score) + ', ' + str(dm_score) + ', '
+ str(cv_score) + ', ' + str(
        bio_score) + ', ' + str(ai_score))
```

6.2.4.2 Consider the probability of given abstract for all 6 classes separately (Method 2)

Second method is getting the abstracts one by one and considers the probability of given abstract for all 6 classes separately. After that if the probability is greater than or equal to 0.5, abstract will be added to that class. Implementation of that part as follows. (Results of this method is included in Appendix B)

```
def predict interests(abstracts, u name, u id):

    model_file = 'model/ieee_SVC_kernal=sigmoid_prob.sav'
    # model_file = 'model/ieee_SVC_kernal=rbf_g05_prob.sav'
    loaded model = pickle.load(open(model file, 'rb'))

    ml = []
    bd = []
    dm = []
    cv = []
    bioin = []
    ai = []

    all_prob = []

    for abstract in abstracts:
        prob = loaded model.predict proba([abstract])
        all prob.append(prob)

    # print(all_prob)
    limit = 0.49

    for p in all prob:
        # print(p[0][0])
        for i in range(0, 1):
            if(p[0][i] > limit):
                ml.append(p[0][i])
            if(p[0][i + 1] > limit):
                bd.append(p[0][i + 1])
            if (p[0][i + 2] > limit):
                dm.append(p[0][i + 2])
            if (p[0][i + 3] > limit):
                cv.append(p[0][i + 3])
            if (p[0][i + 4] > limit):
                bioin.append(p[0][i + 4])
            if (p[0][i + 5] > limit):
                ai.append(p[0][i + 5])

    ml_len = ml.__len__()
    bd_len = bd.__len__()
    dm_len = dm.__len__()
    cv_len = cv.__len__()
    bio_len = bioin.__len__()
    ai_len = ai.__len__()

    denominator = ml_len + bd_len + dm_len + cv_len + bio_len + ai_len

    ml_score = round(((ml_len / denominator) * 100), 2)
    bd_score = round(((bd_len / denominator) * 100), 2)
    dm_score = round(((dm_len / denominator) * 100), 2)
    cv_score = round(((cv_len / denominator) * 100), 2)
    bio_score = round(((bio_len / denominator) * 100), 2)
    ai_score = round(((ai_len / denominator) * 100), 2)

    print(str(ml score) + ', ' + str(bd score) + ', ' + str(dm score) + ', ' +
str(cv score) + ', ' + str(bio score)
          + ', ' + str(ai_score))

    with open('users/users.csv', mode='a', encoding="utf-8", newline='') as file:
        writer = csv.writer(file, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
        writer.writerow([u id, u name, ml score, bd score, dm score, cv score,
bio_score, ai_score])
```

6.2.5 Topic Modeling Approach (Method 3)

Topic modeling is another interesting unsupervised technique to find a solution to profiling users. In here Latent Semantic Analysis approach is used to generate the user profiles.

In LSA approach 6 different corpuses are being created for our 6 selected areas. Each corpus has set of extracted features with their occurrences value. The results can be tested by changing the number of features in the concepts or changing the ngram range of the sentence/ word.

Here is the sample code of the LSA model.

In this code, it gets the 6 datasets and creates 6 different concepts for the selected six different classes. First one file is taken and converts the words to TF-IDF and truncated those. After getting top features as given in to code (sortedTerms = sortedTerms[:200]) , added them into the concept_words dictionary as Concept 1 (In here it is Machine Learning Concept). Like this it iterate over the all the 6 files and create the 6 concepts and stored in the concept_words dictionary.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
import csv
import nltk

files = ['ieee_dataset/machine_learning_dataset.csv',
'ieee_dataset/big_data_dataset.csv', 'ieee_dataset/data_mining_dataset.csv'
    ,'ieee_dataset/computer_vision_dataset.csv',
'ieee_dataset/bioiformatic_dataset.csv',
'ieee_dataset/artificial_intel_dataset.csv']
concept_words = {}
count = 1

for f in files:
    corpus = []
    with open(f, encoding="utf-8") as input_file:
        readCSV = csv.reader(input_file, delimiter=',')
        for row in readCSV:
            corpus.append(row[1])

    vectorizer = TfidfVectorizer(ngram_range=(1, 1))
    X = vectorizer.fit_transform(corpus)
    # print(vectorizer.get_feature_names())
    # print(X[1])
    # print(X.shape)

    lsa = TruncatedSVD(n_components = 1, n_iter = 20)
    lsa.fit(X)

    # Visualizing the concepts
    terms = vectorizer.get_feature_names()
    for i, comp in enumerate(lsa.components_):
        componentTerms = zip(terms,comp)
        sortedTerms = sorted(componentTerms,key=lambda x:x[1],reverse=True)
```

```
        sortedTerms = sortedTerms[:200]
        concept_words["Concept "+str(count)] = sortedTerms
    count = count + 1
    # print(concept_words)
print(concept_words)
```

In the first step it could not be decided what the features should include in each concept. So this is an unsupervised learning method.

After creating the defined concepts, (in here, number of concepts = 6), the user's abstracts are being taken as before methods and preprocessed those. After that each abstract is being iterated through all 6 concepts. Then it generate 6 scores for 6 concepts based on features similarity. After that system get the highest valued concept as the relevant category for that abstract. After iterate all the abstracts of the logged user through those concepts, system generated the user profile as % for each categories. (Results are included in Appendix B)

```
test = []

with open('users/user_file.csv', encoding="utf-8") as input_file:
    readCSV = csv.reader(input file, delimiter=',')
    for row in readCSV:
        test.append(row[1])

limit = test.  len  ()
print(limit)
output = []
all scores = []
# Sentence Concepts
for key in concept_words.keys():

    sentence scores = []

    for sentence in test:
        words = nltk.word_tokenize(sentence)
        score = 0
        for word in words:
            for word with score in concept words[key]:
                if word == word_with_score[0]:
                    score += word_with_score[1]
        sentence_scores.append(score)
        all scores.append(score)
    print("\n"+key+":")

    sum = 0
    for sentence_score in sentence_scores:
        print(sentence_score)
        sum = sum + sentence score
    avg = (sum / limit) * 2
    output.append(avg)

ml = []
bd = []
dm = []
cv = []
bio = []
ai = []

count = 1
for score in all scores:
    if(count <= (results len/6)*1):
        ml.append(score)
        count = count + 1
```

```
    elif(count <= (results_len/6)*2):
        bd.append(score)
        count = count + 1
    elif(count <= (results len/6)*3):
        dm.append(score)
        count = count + 1
    elif (count <= (results_len / 6) * 4):
        cv.append(score)
        count = count + 1
    elif (count <= (results len / 6) * 5):
        bio.append(score)
        count = count + 1
    elif (count <= (results_len / 6) * 6):
        ai.append(score)
        count = count + 1
    else:
        print('Process finished with error')

a = []
b = []
c = []
d = []
e = []
f = []

for i in range(0, int(results len/6)):
    # if(ml[i] == 0 and bd[i] == 0 and img[i] == 0):
    #     break

    if(ml[i] > bd[i] and ml[i] > dm[i] and ml[i] > cv[i] and ml[i] > bio[i] and ml[i]
> ai[i]):
        a.append(ml[i])

    elif(bd[i] > ml[i] and bd[i] > dm[i] and bd[i] > cv[i] and bd[i] > bio[i] and
bd[i] > ai[i]):
        b.append(bd[i])

    elif (dm[i] > ml[i] and dm[i] > bd[i] and dm[i] > cv[i] and dm[i] > bio[i] and
dm[i] > ai[i]):
        b.append(bd[i])

    elif(cv[i] > ml[i] and cv[i] > bd[i] and cv[i] > dm[i] and cv[i] > bio[i] and
cv[i] > ai[i]):
        d.append(cv[i])

    elif (bio[i] > ml[i] and bio[i] > bd[i] and bio[i] > dm[i] and bio[i] > cv[i] and
bio[i] > ai[i]):
        e.append(cv[i])

    elif (ai[i] > ml[i] and ai[i] > bd[i] and ai[i] > dm[i] and ai[i] > cv[i] and
ai[i] > bio[i]):
        d.append(cv[i])

ml_len = (len(a))
bd_len = (len(b))
dm_len = (len(c))
cv_len = (len(d))
bio_len = (len(e))
ai_len = len(f)

print(ml_len)
print(bd_len)
print(dm_len)
print(cv_len)
print(bio_len)
print(ai_len)

ml_score = (ml_len / int(results_len/6)) * 100
bd_score = (bd_len / int(results_len/6)) * 100
dm_score = (dm_len / int(results_len/6)) * 100
cv_score = (cv_len / int(results_len/6)) * 100
bio_score = (bio_len / int(results_len/6)) * 100
ai_score = (ai_len / int(results_len/6)) * 100

print(str(ml_score) + ', ' + str(bd_score) + ', ' + str(dm_score) + ', ' +
str(cv_score) + ', ' + str(bio_score) + ', ' + str(ai_score))
```

6.3 Recommendation of journals for manuscript module

Data gathering

In here it is being used two inputs of training dataset and testing dataset. Manually created training dataset used for detect theme for given manuscript. Other dataset is journals details and submitting manuscript. Journals data provided by springerOpen[1]. It has 200 more journals with different categories.

Extract contain of manuscript

In this step has included text extraction from manuscript. I used pdf to doc python library for archive this task. Mainly focus title, abstract and keywords of the article. Manuscript is uploaded from angular front end and save to flask backend.finaly user can extract the document then necessary details will push to array.

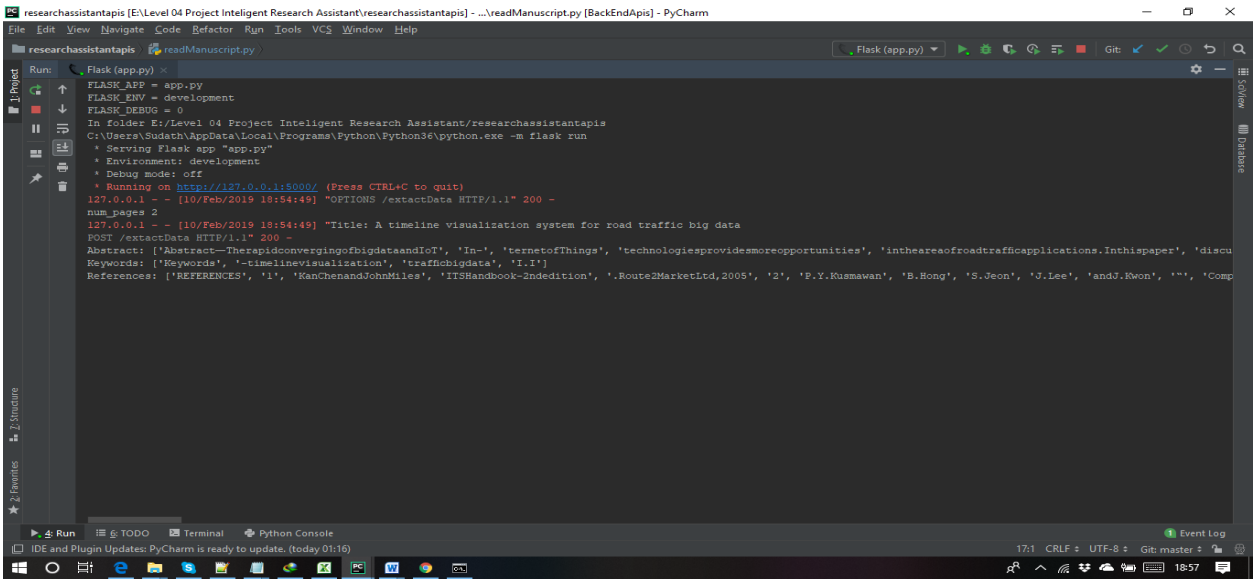


Figure 6.6: Data extracting from manuscript

Web scraping

For web scraping first import the two scraping libraries of the Urllib and Beutifusoup. After the receiving the data source url then scraping the journal data relate to the request. Through the Urllib reach to the given Urls of websites and read the data of it. First split the url and identify the webpage name and remove strings like journal, articles, etc. here determine if any words of above mentioned are included in url, it is not a journal. After that read the heading and content of the article.

After reading the article remove the HTML and XML tags and process the article via beautifulsoup libraray. Here remove header tags of <h1>,<h2>…tags and other tags like <a>,<href> tags. From removing these tags get only text of the article.

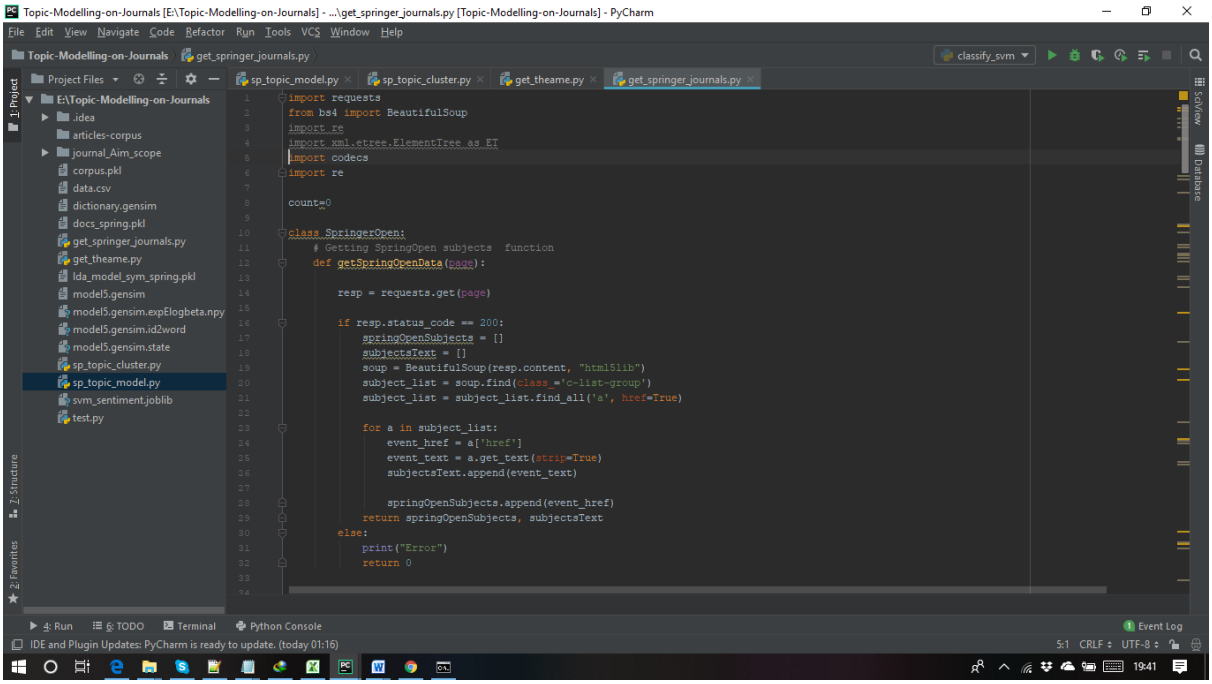


Figure 6.7: Web scrapping from journals

Data Pre-processing

After scraping the data need pre-process, them in structured way. First create the libraries of "cleaner and re" libraries. Through the cleaner library remove the navigations bars, headers and footers of the articles, comments of the article, removing stop words of do not have any meaning like "a,an, the, is, "and so on. Using "re" library and develop the regular expressions for remove punctuations, extraneous symbols like '#,@,*', ignore cases and convert into same case, remove numbers, remove white spaces, expanding abbreviations etc.

Further to pre-process the data import the NLTK python library. Through that able to tokenize, stemming, lemmatize, tagging, parsing and classification the articles.

By using nltk. wordTokenizer library able tokenize the whole content in to words. It is easy to classify the data. Then using the nltk.stemmer able to stemming the words. That means remove the all affixes (prefixes, suffixes) words and get one word. Then used the nltk.lemmetizer remove the all the canonical word and get on word. Ex: best to good, worst to bad.

Data pre-processing part is used even in manuscript data extraction part. Because we need to remove unnecessary data.

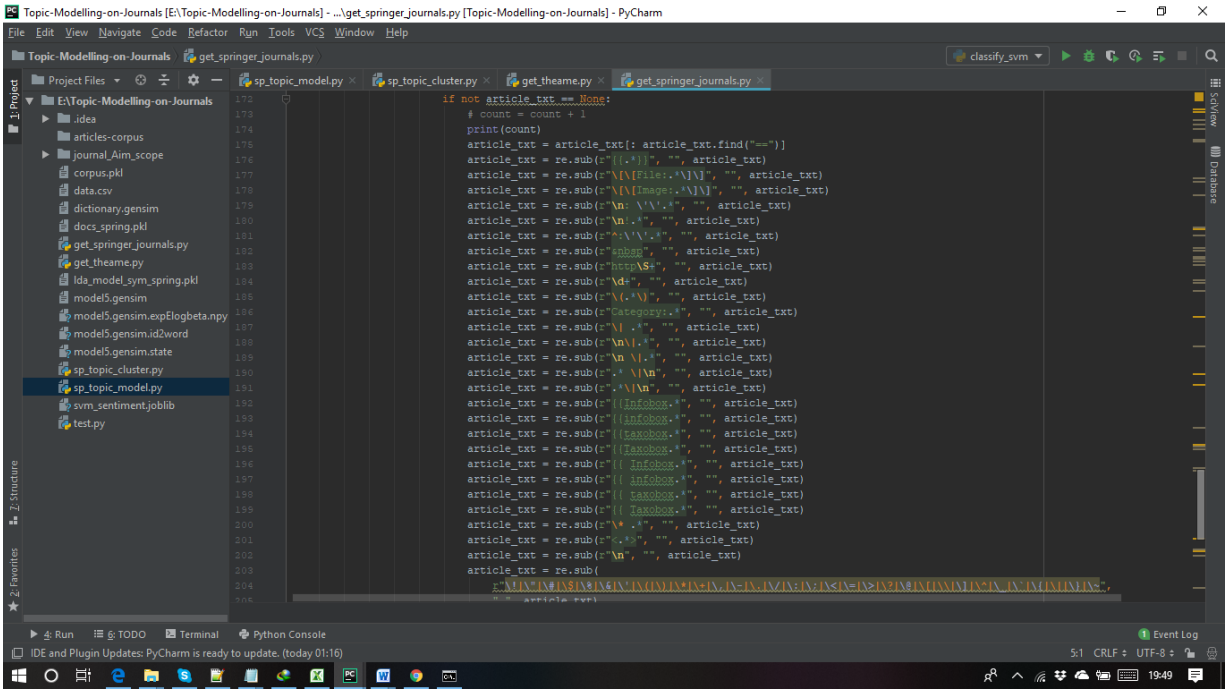


Figure 6.8: Data pre-processing

Building word dictionary

In this step is going to build dictionary and generate all word with ids. Corpus gensim algorithmic was used added two additional steps:finaly see “pkl” file (e.g.: docs_spring.pkl)

1. Every one of the tokens in the lexicon which either have happened in under 2 articles or have happened in over 40% of the articles are expelled from the lexicon, as these words won't contribute the different subjects or points.

2.Second thing is after getting the most frequent set of words from the dictionary, can be found what are the most related words for the article.

```
corpus_path ="journal_Aim_scope/"
article_paths = [os.path.join(corpus_path,p) for p in os.listdir(corpus_path)]

# Read contents of all the articles in a list "doc_complete"
doc_complete = []

for path in article_paths:
fp = codecs.open(path,'r','utf-8')
doc_content = fp.read()
doc_complete.append(doc_content)

# Randomly sample 140 articles from the corpus created from the 1st blog-post (wiki_parser.py)
docs_all = random.sample(doc_complete, 140)
docs =open("docs_spring.pkl",'wb')
cPickle.dump(docs_all, docs)

# Use 150 articles for training.
docs_train = docs_all[:140]
# docs_train = docs_all[:len(docs_all)]
# Cleaning all the 150 simplewiki articles
stop =set(stopwords.words('english'))
lemma = WordNetLemmatizer()
doc_clean = [clean(doc) for doc in docs_train]
```

Figure 6.9: algorithm of dictionary generate

Feature extraction (Bag of words)

This gives the knowledge that comparative reports will have word tallies like one another. At the end of the day, the more comparable the words in two archives, the more comparative the reports can be. This will be generate word count vector for the dictionary.

LDA Model Training

In this step can see words with frequency counts. Also it will be created lda model pkl file (“lda_model_sym_spring.pkl”) file. All words belong to the some topic and this time generated 50 topics and one topic has 10 words.

```
#Creating the object for LDA model using gensim library & Training LDA model on the document term matrix.
ldamodel = Lda(doc_term_matrix, num_topics=50, id2word = dictionary, passes=50, iterations=500)
ldafile = open('lda_model_sym_spring.pkl','wb')
cPickle.dump(ldamodel,ldafile)
ldafile.close()

#Print all the 50 topics
for topic in ldamodel.print_topics(num_topics=50, num_words=10):
print(topic[0]+1, " ", topic[1],"\n")
```

Figure 6.10: Algorithm of LDA topic

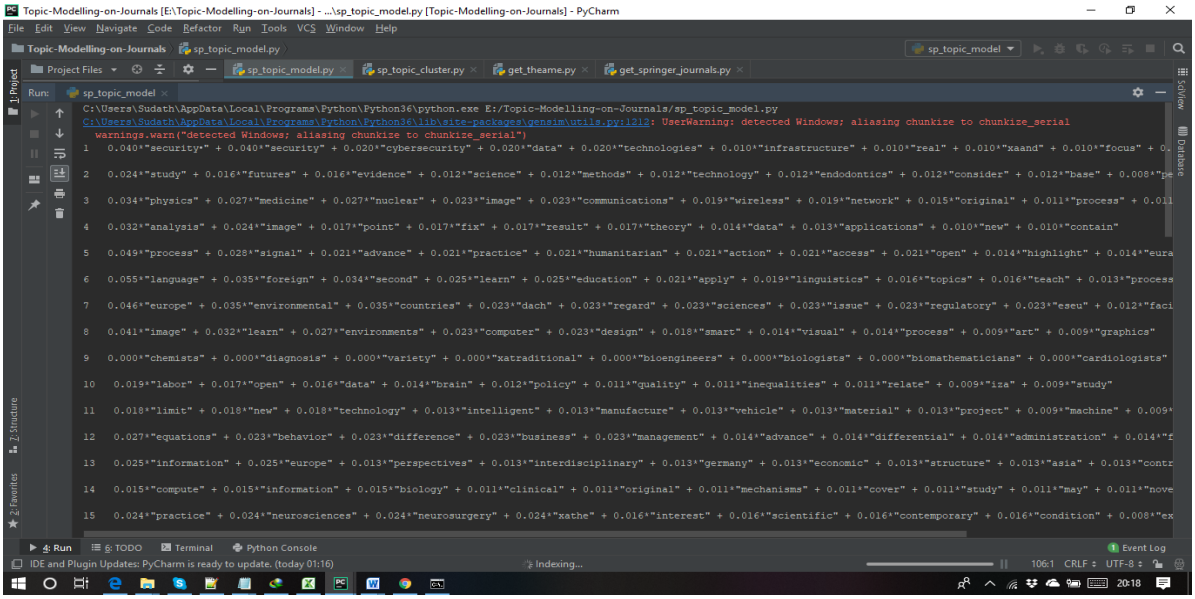


Figure 6.11: Result of LDA topic

Theme extraction from manuscript (combination of LDA and SVM models)

This is a hybrid module. After submitting the manuscript, system need to identify field of submitted paper, therefore system has to automatically identify the related area of paper.

I used LDA model to generate topics and their frequency counts. But it cannot labels there for I used again SVM module to identify relevant theme. Also after training the dataset can be seen accuracy rate. It has 76% accuracy. Also it will be generated this file. (“svm_sentiment.joblib”).

Next thing is generating correct theme for the manuscript. Then svm algorithm will send topic list for the svm model and will be output related theme.

```
def test_and_train(self, sentences, vectorized_data):
print ('training')
    indexed_data = hstack((np.array(range(0, vectorized_data.shape[0]))[:, None], vectorized_data))

def sentiment2target(sentiment):
return {
'database': 0,
'medicine': 1,
'engineering': 2
}[sentiment]

    targets = sentences.company_sentiment.apply(sentiment2target)
    data_train, data_test, targets_train, targets_test = train_test_split(indexed_data, targets, test_size=0.4,
random_state=0)
data_train_index = data_train[:, 0]
    data_train = data_train[:, 1:]
data_test_index = data_test[:, 0]
    data_test = data_test[:, 1:]
    clf = OneVsRestClassifier(
        svm.SVC(gamma=0.01, C=100., probability=True, class_weight='balanced', kernel='linear'))
clf_output = clf.fit(data_train, targets_train)
    score = clf.score(data_test, targets_test)
print(score)
    dump(clf, 'svm_sentiment.joblib')
return clf
```

Figure 6.12: Algorithm of svm train and testing

```
preprocess called
['database big data relational datamining.']
database      9180
medicine      3096
engineering   2364
Name: company_sentiment, dtype: int64 14640
training
0.7662226775956285
svm model creating

[{
'comment': 'database big data relational datamining.',
 'database': 0.7917072280810592,
 'medicine': 0.13018867531084857,
 'engineering': 0.07810409660809209
}]
```

Figure 6.13: Result of svm model

6.4 The connectivity detection module

Data collection

Dataset which built from Google Scholar with the author name, co-authors and c-authors of co-authors names, author's affiliation, research paper title are extracted via python code. The data source of the approach is Google Scholar. Scholarly is a library that allows to retrieve author and publication information from Google Scholar in a Pythonic way. The number of research paper titles to gather data is also given. [16].

Data preprocessing

Correct and detailed data is the main key point to build a reliable dataset. There were some difficulties in data collecting process. One thing is the author name should be the same in each and every research paper. If author name different due to the spaces or case of the letter that author name takes place as another new node in the graph. As a solution for that remove spaces that containing two sides of the name and all names are retrieved in lower case. When search an author name in Google Scholar, sometimes displays several authors with same the scholary unable to identify the exactly related author. Hence to prevent that need to match the titles of the author with co-author's name that have co-authored. Another thing is co-author names that include in the research papers but not in the Google Scholar is also ignored when gathering data.

[11] [12].

Here is the python code for retrieve data from Google Scholar.

```
import scholarly
import pymongo
import pandas as pd
import re

client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client["Scholar"]
author_col = db["Author_S_and_co"]
data_col = db["Author_S_and_co_data"]
data_col.delete_many({})

author_df = pd.DataFrame(list(author_col.find()))

def get_author(author_string):
    try:
        search_query = scholarly.search_author(author_string)
        author = next(search_query).fill()
        author_name = author.name.lower()
        affiliation = author.affiliation
        title = []
        co_authors = []

       for i , pub in enumerate(author.publications):
            if i == 10:
                break
            pub = pub.fill()
            title.append(pub.bib.get('title'))
            temp_co_authors = ''
            if (pub.bib.get('author') != None):
                temp_co_authors = pub.bib.get('author').lower()
                 temp_co_authors = temp_co_authors.replace(author_name
+ ' and', '')
                   temp_co_authors = temp_co_authors.replace('and ' +
author_name, '')
                                                        temp_co_authors    =
temp_co_authors.replace(author_name, '')
                if (temp_co_authors.count('and') >= 1):
                    temp_co_authors = temp_co_authors.split('and')
            if (isinstance(temp_co_authors, list)):
                for i, val in enumerate(temp_co_authors):
                      temp_co_authors[i] = re.sub('(^\s*)|([ \s]+$)',
'', val)
```

```
            co_authors.append(temp_co_authors)
            print("Title " + str(i+1) + " completed")

                    Author_and_co_data = [{"Author": author_name,
"affiliation": affiliation, "title": title,"co_authors":co_authors}]
        data_col.insert_many(Author_and_co_data)
    except:
        print(str(author_string)+" not found in google scolar")


for i , author in enumerate(author_df['Author']):
    get_author(author)
    print("Author "+str(i+1)+" completed")
```

Figure 6.14: Code segment for retrieving data from Google Scholar

Affiliation Network

The cypher query for visualization of the Affiliation network is as shown below,

```
CALL apoc.load.json("file:C:/Affiliation_Data.json")YIELD value AS Data
MERGE (A:Author{Author_name:Data.Author})
MERGE (Af:Affiliation{Author_affiliation:Data.affiliation})
CREATE (A)-[r:IS_AFFILIATE_TO]->(Af)
RETURN A,Af,r
```

Figure 6.15: Cypher query for Affiliation network

Here use apoc library to load json file that contained data set which needed to build graph.

Now this obtained Author name and Affiliation is used to match create relationship as IS_AFFILIATE_TO between those nodes.

Figure 6.16: Section of Affiliation Network containing authors in orange and affiliation in blue
(Authors are representing in orange and affiliation is representing in blue.)

The purpose of this approach is giving a prediction to the researchers to find out collaborators in the particular affiliation. It is not an easy task to find collaborators for further works. However, there is a high chance to become as collaborators due to the physical interaction, if user's research areas are matched. In other hands If the researcher prefers to move to the new research area, it is also difficult to find out collaborators due to lack of knowledge about the new area. It is valuable and easy to acquire knowledge and experience if collaborators are in the same place.

The cypher query for matching required affiliation is below,

```
START Author=node(*)
MATCH (Affiliation{Author_affiliation:'University of Moratuwa'})-[r:IS_AFFILIATE_TO]-(Author)
RETURN Author, Affiliation, Author.Author_name as Authors
```

Figure 6.17: Cypher query segment of matching required affiliation

The output of the query is visualize graph is as below,

Figure 6.18: Visualization of query

The table of the Table of the listed authors in same affiliation is as below,

"Authors"
"thushari silva"
"sagara sumathipala"
"supunmali ahangama"
"dr. g. upeksha ganegoda"
"dr. lochandaka ranathunga"
"asoka karunananda (orcid:0000-0002-2458-6195)"
"mohamed fazil mohamed firdhous"
"dr. subha fernando"
"prabhashrini dhanushika"
"sarada hettiarachchi"

Figure 6.19: Table of the listed authors in same affiliation

Above table provides authors names that related to the user's affiliation which the University of Moratuwa. With the matching of the research areas that the authors are engaged is to provide collaborator recommendation. The matching of the preferred area with the predicted authors who are in the same affiliation is done by the collaborator recommendation module

Co-Authorship Network

Visualization of the Co-Authorship Network is done in neo4j browser using cypher query. The purpose of this approach is provided the strongest connected co-authors and strongest co-authors of co-authors. Here, the number of research paper titles that have co-authored with the user is used to measure the level of strength of the relationship between them. Researchers need collaborators to do their future research works. So that using the co-authorship network able to suggest the strongest connected co-author. But those co-authors and the user already have some connection. It may not be due to that in this approach provide the suggestion of the co-authors of co-authors that have a strong connection. The Co-Authorship Network that shows in below visualize those two connections [26].

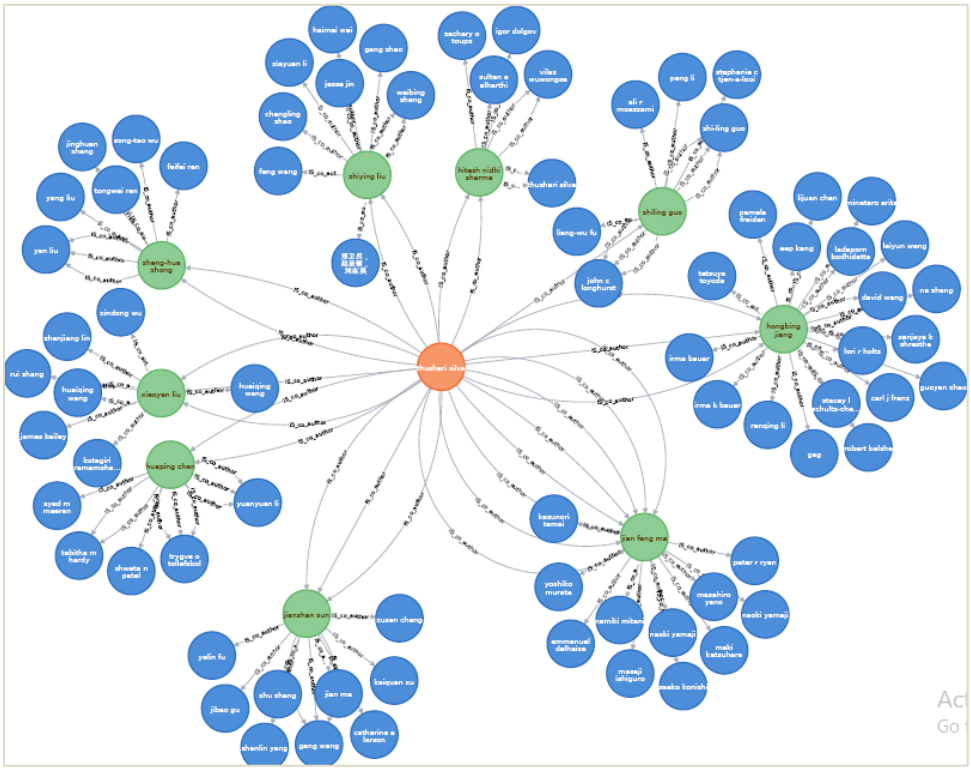


Figure 6.20: The co-authorship network containing one authors and ten coauthors (The authors are represented in red and co-authors in blue)

The cypher query for create ontology of co-authorship network is below,

```
CALL apoc.load.json("file:C:/Data.json")YIELD value AS Data
MERGE (A:Author{Author_name:Data.Author})
FOREACH (co_author_arr IN Data.co_authors |
        FOREACH (co_authorName IN co_author_arr |
                MERGE (C:co_Author{co_author_name:co_authorName})
                CREATE (A)-[R1:IS_co_author]->(C)
        )
)
RETURN *
```

```
FOREACH (co_author_obj IN Data.co_authors_details |
        MERGE (C:co_Author{co_author_name:co_author_obj.Author})
        FOREACH (co_co_author_arr IN co_author_obj.co_authors |
                FOREACH (co_co_author IN co_co_author_arr |
                        MERGE (CC:co_co_Author{co_co_author_name:co_co_author})
                        CREATE (C)-[R2:IS_co_author]->(CC)
                )
        )
)
RETURN *
```

Figure 6.21: Cypher query of creating ontology of co-authorship network

The cypher query for count relationships between author and co-authors as follows,

```
MATCH (C:co_Author)-[R2:IS_co_author]-(A:Author)
RETURN C.co_author_name as co_author, count(R2) as rel_count
ORDER BY rel_count DESC;
```

Figure 6.22: Cypher code of counting the number of relationships with co-authors

$$RC = \sum R1$$

RC means the relationship count of the each co-authors. R1 is the relationship count between the author and the co-author RC is equal to the summation of the relationship count (R1)

Below table shows that graphical representation in details

"co_author"	"rel_count"
"jian ma"	8
"xiaoyan liu"	3
"jianshan sun"	3
"hongbing jiang"	3
"sheng-hua zhong"	2
"zhiying liu"	2
"huaping chen"	2
"hitesh nidhi sharma"	2
"mingyu zhang"	1
"zhiling guo"	1

Figure 6.23: Table of co-authors of one author

Above table shows the relationship count of connected co-authors that the Author has co-authored. According to the above, the sample graph Jian Ma is the most connected co-author is named Thushari Silva. So, using these details can identify the strongest connected co-authors.

Then have to identify that the co-authors of co-authors analyzing the graph.

```
MATCH  (A1:Author)-[R1:IS_co_author]->(C1:co_Author)
WITH A1,C1,count(R1) as rel_count1
WITH A1,collect(C1) as co_authors,rel_count1
UNWIND co_authors as co_author
MATCH (co_author)-[R2:IS_co_author]->(co_co_Author)
WITH co_author,rel_count1,co_co_Author,count(R2) as rel_count2
WITH co_author,rel_count1,co_co_Author,rel_count2,rel_count1+rel_count2 as
total,(rel_count1*2)+(rel_count2*1) as weightedScore
RETURN co_author.co_author_name as co_author,rel_count1,co_co_Author.co_co_author_name  as
co_co_Author,rel_count2,total,weightedScore
ORDER BY weightedScore DESC;
```

Figure 6.24: cypher query for counting an identifying co-authors of co-authors

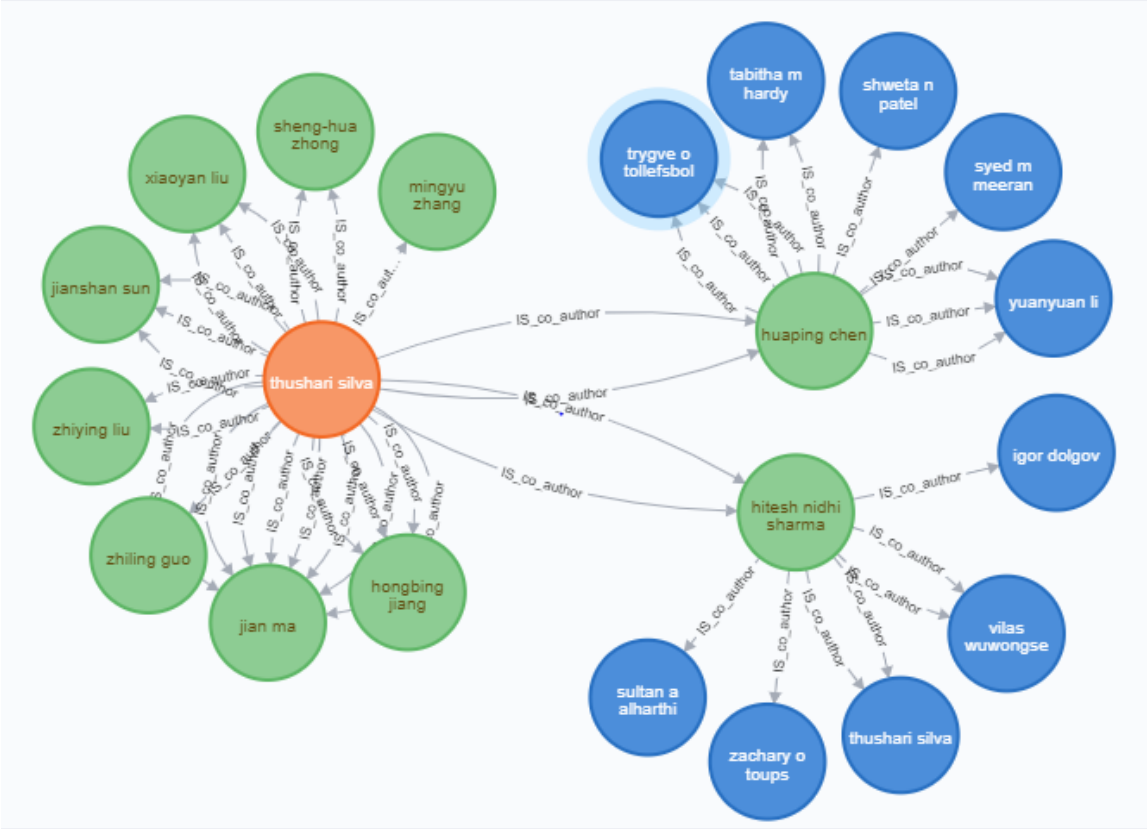


Figure 6.25: Section of the Co-Authorship network which visualize the relationship of co-authors of co-authors

The highest value of the relationship count between co-author and the co-author of the co-author is not predicted the strongest connected co-authors of co-authors. Because of the relationship between the author and co-author has more strength than it. Below table shows that author gang wang rel_count is 3 and between the author co-author relationship with author jianshan sun (rel_count1) is also three. Even though Shidun Cheng's rel_count2 is 1,rel_count2 of the Shidun Cheng with jian ma is 8. Therefore author Shidun Cheng connection has more strength than the author gang wang. It meat that the rel_count1 has more strength. So here used a weighted score to get the corrected values.

$$Weighted\ Score = \sum R1*W1 + \sum R2*W2$$

R1 is the relationship count between the author and the co-author. R2 is the relationship count between the co-authors and the co-authors of co-authors. W1 and W2 is the given weight for the first relationship and second relationship. W1 value is 2 and W2 value is 1. According to that Shidun Cheng and ten other authors become

the top. Because they all are co-authors of the co-author named Jian ma who has the strongest connection with the author names Thushari Silva.

"co_author"	"rel_count1"	"co_co_Author"	"rel_count2"	"total"	"weightedScore"
"jian ma"	8	"Shiduan Cheng"	1	9	17
"jian ma"	8	"Clara Deser"	1	9	17
"jian ma"	8	"Andrew T "	1	9	17
"jian ma"	8	"Haitao Wu"	1	9	17
"jian ma"	8	"Shang-Ping Xie"	1	9	17
"jian ma"	8	"Haiyan Teng"	1	9	17
"jian ma"	8	"Gabriel A Vecchi"	1	9	17
"jian ma"	8	"Keping Long"	1	9	17
"jian ma"	8	"Yong Peng"	1	9	17
"jian ma"	8	" Zhi-Ping Fan"	1	9	17
"jianshan sun"	3	"gang wang"	3	6	9
"hongbing jiang"	3	"david wang"	2	5	8

Figure 6.26: Segment of the table of co-authors of co-authors with relationship count

6.5 Collaborators Recommendation Module

In collaborators Recommendation module, when a new user is logged the collaborators for him for further work are being recommended based on relativity to the most interested subject area and social connectivity. First when a new user is logged the profiling module is giving the weight he has for each subject area separately. For the recommendation module the main input is that.

```
logged_user = ['0051', 'user 51', 10, 6, 13, 50, 15, 16]
cal_pearson_correlation(logged_user)
```

Figure 6.27: Entering new user values

After that the new user will be compared with the existing users one by one and top 10 users are being selected using the spearman's correlation algorithm.

```
def cal_spearman(new_user):
    ranks = {}

    # get all user details in the system
    users = get_all_users()

    # cal spearman correlation for each user with logged user
    for count in range(1, users.__len__()):

        u_name = users[count][0]
        u_ml = float(users[count][1])
        u_bd = float(users[count][2])
        u_dm = float(users[count][3])
        u_cv = float(users[count][4])
        u_bio = float(users[count][5])
        u_ai = float(users[count][6])

        d_ml = math.pow((new_user[1] - u_ml), 2)
        d_bd = math.pow((new_user[2] - u_bd), 2)
        d_dm = math.pow((new_user[3] - u_dm), 2)
        d_cv = math.pow((new_user[4] - u_cv), 2)
        d_bio = math.pow((new_user[5] - u_bio), 2)
        d_ai = math.pow((new_user[6] - u_ai), 2)

        sum = d_ml + d_bd + d_dm + d_cv + d_bio + d_ai

        r = 10 - (6 * sum) / 210
        ranks[u_name] = r
```

Figure 6.28: Using Spearman's correlation algorithm

Now the correlation values of the users are being selected. After that the users who have the correlation value 7 are being selected by using a cutoff mark because the objective is to recommend the highly related collaborators.

```
def get_connected_users():

    connective_users = []

    # append the user details received from the user connectivity module into array
    with open('testdataconnectivity.csv') as file:
        read_file = csv.reader(file, delimiter=',')

        for row in read_file:
            connective_users.append(row)

    # return the 2D array of connective users
    return connective_users


def get_top_correlated_users(correlation_values, cut_of_r):

    sorted_set = nlargest(correlation_values.__len__(), correlation_values, key=correlation_values.get)

    r = []

    # get user names has correlation value from 7 (r)
    if(cut_of_r >= 7):
        for name in sorted_set:
            if (correlation_values.get(name) >= cut_of_r):
                r.append(name)
        return r
    else:
        return 0
```

Figure 6.29: Get correlated users

In spearman's correlation algorithm, the connectivity will have to be more than 0 but it doesn't have to be a higher value because a person which has at least a small connectivity value should be recommended as assumed. Then the connectivity value will be gained by the social connections module.

```
def find_common_set(correlation_values, r):
    # get connective users set calling get_connected_users()
    connectivity = get_connected_users()

    # declare an array to store users name and connectivity score, if connectivity score >= cut_of_score(c)
    c = []
    for num in range(1, connectivity.__len__()):
        # if (float(connectivity[num][1]) >= cut_of_score):
        c.append(connectivity[num])
    # print(c)

    # define an array to store common set
    common_set = []

    # find the common set
    for num in range(0, c.__len__()):

        user = c[num][0]
```

Figure 6.30: Get connected users

The common sets of users are being extracted based on above mentioned two data sets. In Fuzzy String matching algorithm the users who get 100 of matching the IDs of the users will be entered to the common set and the other users are also being entered but they will be remain in the back since they have less than 100 value.

```
def fuzzy_similarity(query, choices):
    # input the values into fuzzy string matching algorithm
    result = process.extract(query, choices, limit=1)
    return result
```

Figure 6.31: Using Fuzzy logic

```
def find_common_set(correlation_values, cut_of_r, cut_off_c, r, u_id_with_name):

    # get connective users set calling get_connected_users()
    connectivity = get_connected_users()

    # declare an array to store users name and connectivity score, if connectivity score >= cut_of_score(c)
    c = []

    for num in range(1, connectivity.__len__()):
        if(float(connectivity[num][1]) >= cut_off_c):
            c.append(connectivity[num])
    print(c)

    # define an array to store common set
    common_set = []

    r_id = []

    for name in r:
        r_id.append(u_id_with_name.get(name))

    # find the common set
    for num in range(0, c.__len__()):

        user = c[num][0]
        id = u_id_with_name.get(user)

        # get output of the matching algorithm
        fuzzy_output = fuzzy_similarity(id, r_id)
        print(fuzzy_output)
```

Figure 6.32: Get the common set of users

Then the recommendation part can be done because the common set is also there. From the common set the best collaborators will be selected after adding the c value and r value and ordering them to the descending order.

```
def final_recommendation(common_set, correlation_values, c, cut_off_c, cut_of_r, r, u_id_with_name):

    recommendation = {}

    print('common set')
    print(common_set)

    common_set_names = []
    for id in common_set:
        name = [key for (key, value) in u_id_with_name.items() if value == id]
        common_set_names.append(name)

    print(common_set_names)

    # find the length of common set
    length_common_set = len(common_set)

    common_set = []
    for i in range(0, length_common_set):
        common_set.append(common_set_names[i][0])

    # get the recommendation limit (no of users should recommend)
    limit = no_of_recommendation()

    # create a set of c
    c_dictionary = {}
    for i in range(0, c.__len__()):
        c_dictionary[c[i][0]] = float(c[i][1])

    # look if we can do the recommendation using only common set
    if(length_common_set >= limit):
        # loop through the username one by one in common set
```

Figure 6.33: Final recommendation 1

```
# look if we can do the recommendation using only common set
if(length_common_set >= limit):
    # loop through the username one by one in common set
    for u_name in common_set:
        # declare and assign value of user's correlation value
        r_value = correlation_values.get(u_name)

        # declare and assign value of user's connectivity value
        c_value = c_dictionary.get(u_name)

        # get common value using r_value and c_value
        common_value = r_value + c_value

        # store the common value with username in dictionary
        recommendation[u_name] = common_value

    # find most matching users as user want
    sorted_recommendation = nlargest(limit, recommendation, key=recommendation.get)
    print('Top ' + str(limit) + ' recommendation based on user profile')
    for user in sorted_recommendation:
        print(user + ' : ' + str(recommendation.get(user)))

else:
    cut_off_c = cut_off_c - 1
    if(cut_off_c > 2):
        print('------------------------>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>')
        print('cutoff score for c--------')
        print(cut_off_c)
        find_common_set(correlation_values, cut_of_r, cut_off_c, r, u_id_with_name)
    else:
        print('Sorry, we cannot recommend the no of users you want. Please reduce the limit you entered')
```

Figure 6.34: Final recommendation 2

The system automatically provides 5 collaborators and if the user wants to find more users he can add the number of users they want to find and the system will suggest the users if available. If not the system will reduce the cutoff marks and try to give the number of users however. If cannot recommend the users following message will be displayed.

[['User 10'], ['User 20'], ['User 26']]

Sorry, we cannot recommend the no: of users you want. Please reduce the limit you entered

Figure 6.35: message of lack of collaborators

When using the Pearson's correlation algorithm, the relevance value and connectivity value both had limits and only the adding algorithm part is being changed. The output of the spearman's algorithm will be displayed below.

```
User 16 : -63.2
['User 10', 'User 16', 'User 18', 'User 20', 'User 26']
Top 5 recommendation based on user profile
User 10 : 19.0
User 26 : 7.0285714285714285
User 20 : 2.257142857142858
User 18 : -51.17142857142858
User 16 : -57.2

Process finished with exit code 0
```

Figure 6.36: Output of Pearson's Correlation Algorithm

6.6 Summary

The chapter six is described the implementation of all four modules with code segments/ algorithms, as well as the necessary steps and outputs. It is provided comprehensive understanding about how the system is built with combining different modules.

Chapter 07

Evaluation

7.1 Introduction

Previous chapter discussed about the analysis and design of each module in the system. This chapter will be included the experimental designs, selection of participants, control experiments, interview techniques, design of questionnaire and further details. Finally the statement whether the objectives are achieved will be mentioned here.

7.2 Evaluation of the profiling Module

In this system, when a new user or researcher logs into the system the Google Scholar name of the particular user is being asked and the system automatically gets 100 abstracts from the most cited publications published by the user in Google Scholar. Then by using those abstracts the system decides the user interest areas based on different approaches.

In this research selected areas were limited to 3 categories as Machine Learning(ML), Big Data(BD) and Image Processing(IMG) as the first level classification. After that it was increased it to 6 as Machine Learning (ML), Big data (BD), Data mining (DM), Computer vision (CV), Bioinformatics (BIO), and Artificial Intelligent (AI).

For topic modeling six data sets are being created for each previously chosen subject area. In here the abstracts which belong to each subject area are being put separately in three files using preprocessing.

Ex: - File 1 - Machine Learning related abstracts
File 2 - Big Data related abstracts
File 3 – Data Mining Related abstracts
File 4 – Computer Vision Related abstracts
File 5 – Bioinformatics Related abstracts
File 6 – Artificial Intelligent Related abstracts

For each file which contains the abstracts, six corpuses are being created including top 50, top 100 and top 200 best related words for each subject area with weights which were feature extracted using TF/ IDF. When a new user logs into the system, 100 abstracts are being taken as mentioned above and here each of the abstracts are being analyzed with each of the concepts and get the values of user interests.

To evaluate different classifiers used, and to select the best classifier for doing profiling, confusion matrixes and classifications reports are being taken of used classifiers with changing the different parameters. After that f1-score and weighted average of those classifiers are being compared and selected the best one. Some of the results of used classifiers as follows.

Confusion Matrix and classification report of some of trained models.

SVC (kernel = linear)

1)

```
X_train, X_test, y_train, y_test = train_test_split(data.abstract,
data.label, test_size=0.2)
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', SVC(kernel='linear', C=1))])


Accuracy ---------------->>>>>  91.5625

[[127   4   4   5   1   2]
 [  3 159   5   2   2   0]
 [  4   8 136   0   5   1]
 [  3   1   1 148   0   2]
 [  5   5   7   2 160   1]
 [  1   3   3   1   0 149]]

              precision    recall  f1-score   support
           1       0.89      0.89      0.89       143
           2       0.88      0.93      0.91       171
           3       0.87      0.88      0.88       154
           4       0.94      0.95      0.95       155
           5       0.95      0.89      0.92       180
           6       0.96      0.95      0.96       157
   micro avg       0.92      0.92      0.92       960
   macro avg       0.92      0.92      0.92       960
       avg       0.92     0.92     0.92       960


filename = 'final_models/ieee_SVC_kernal=linear_k1000.sav'
```

SVC kernel = rbf

1)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', SVC(kernel='rbf', gamma=0.5, C=1))])
```

Accuracy ---------->>>>> **90.52083333333333**

```
[[137   6   0   5   0   4]
 [  9 139   5   1   2   3]
 [  4  10 153   1   5   1]
 [  3   2   0 139   0   4]
 [  4   3   5   1 153   1]
 [  3   2   0   6   1 148]]
```

	precision	recall	f1-score	support
1	0.86	0.90	0.88	152
2	0.86	0.87	0.87	159
3	0.94	0.88	0.91	174
4	0.91	0.94	0.92	148
5	0.95	0.92	0.93	167
6	0.92	0.93	0.92	160
micro avg	0.91	0.91	0.91	960
macro avg	0.91	0.91	0.91	960
weighted avg	0.91	0.91	0.91	960

```
filename = 'final_models/ieee_SVC_kernal=rbf_k100_gamma5_c1.sav'
```

2)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', SVC(kernel='rbf', gamma=0.7, C=1))])
```

Accuracy ---------->>>>> **90.52083333333333**

```
[[130   3   1   5   2   3]
 [  4 156   2   1   1   4]
 [  3   8 120   2   5   1]
 [  3   1   0 164   0   1]
 [  4   3  13   3 146   4]
 [  1   4   2   6   1 153]]
```

	precision	recall	f1-score	support
1	0.90	0.90	0.90	144
2	0.89	0.93	0.91	168
3	0.87	0.86	0.87	139
4	0.91	0.97	0.94	169
5	0.94	0.84	0.89	173
6	0.92	0.92	0.92	167
micro avg	0.91	0.91	0.91	960
macro avg	0.90	0.90	0.90	960
weighted avg	0.91	0.91	0.90	960

```
filename = 'final_models/ieee_SVC_kernal=rbf_k100_gamma7_c1.sav'
```

3)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', SVC(kernel='rbf', gamma=0.1, C=1))])
```

Accuracy ---------->>>>> **87.08333333333333**

```
[[136   4   1   9   1   3]
 [  5 138   5   2   2   1]
 [  8  17 122   2   6   3]
 [  3   2   0 146   0   1]
 [ 10   5   6   1 149   3]
 [  9   4   4   7   0 145]]
```

	precision	recall	f1-score	support
1	0.80	0.88	0.84	154
2	0.81	0.90	0.85	153
3	0.88	0.77	0.82	158
4	0.87	0.96	0.92	152
5	0.94	0.86	0.90	174
6	0.93	0.86	0.89	169
micro avg	0.87	0.87	0.87	960
macro avg	0.87	0.87	0.87	960
weighted avg	0.88	0.87	0.87	960

```
filename = 'final_models/ieee_SVC_kernal=rbf_k1000_gamma1_c1.sav'
```

4)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', SVC(kernel='rbf', gamma=0.5, C=1))])
```

Accuracy ---------->>>>> **90.0**

```
[[141   2   1   3   4   5]
 [  4 140   5   4   0   1]
 [  4  10 146   2   6   3]
 [  2   3   0 139   0   3]
 [  5   1   7   1 141   3]
 [  5   3   3   5   1 157]]
```

	precision	recall	f1-score	support
1	0.88	0.90	0.89	156
2	0.88	0.91	0.89	154
3	0.90	0.85	0.88	171
4	0.90	0.95	0.92	147
5	0.93	0.89	0.91	158
6	0.91	0.90	0.91	174
micro avg	0.90	0.90	0.90	960
macro avg	0.90	0.90	0.90	960
weighted avg	0.90	0.90	0.90	960

```
filename = 'final_models/ieee_SVC_kernal=rbf_k1000_gamma5_c1.sav'
```

5)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1),
sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', SVC(kernel='rbf', gamma=0.7, C=1))])
```

Accuracy ---------->>>>> **91.97916666666667**

```
[[143   2   6   4   3   4]
 [  1 157   4   2   2   2]
 [  1   7 146   3   5   2]
 [  0   0   1 130   0   2]
 [  1   3   5   0 166   0]
 [  3   5   1   8   0 141]]
```

	precision	recall	f1-score	support
1	0.96	0.88	0.92	162
2	0.90	0.93	0.92	168
3	0.90	0.89	0.89	164
4	0.88	0.98	0.93	133
5	0.94	0.95	0.95	175
6	0.93	0.89	0.91	158
micro avg	0.92	0.92	0.92	960
macro avg	0.92	0.92	0.92	960
weighted avg	0.92	0.92	0.92	960

```
filename = 'final_models/ieee_SVC_kernal=rbf_k1000_gamma7_c1.sav'
```

6)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', SVC(kernel='rbf', gamma=0.5, C=100))])
```

```
92.1875
[[132   2   5   1   4   4]
 [  2 140   6   0   1   2]
 [  3   9 132   3   7   2]
 [  4   0   0 166   1   1]
 [  1   1   1   2 154   1]
 [  2   4   2   3   1 161]]
```

	precision	recall	f1-score	support
1	0.92	0.89	0.90	148
2	0.90	0.93	0.91	151
3	0.90	0.85	0.87	156
4	0.95	0.97	0.96	172
5	0.92	0.96	0.94	160
6	0.94	0.93	0.94	173
micro avg	0.92	0.92	0.92	960
macro avg	0.92	0.92	0.92	960
weighted avg	0.92	0.92	0.92	960

```
filename = 'final_models/ieee_SVC_kernal=rbf_k100_gamma5_c100.sav'
```

SVC kernel = sigmoid

1)

```
pipeline = Pipeline([('vect',
TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
('chi', SelectKBest(chi2, k=100)), ('clf',
SVC(kernel="sigmoid", gamma=0.5)])
```

Accuracy ----------->>>>> **88.33333333333333**

```
[[151  6  4 10  3  4]
 [ 5 128  2  1  0  0]
 [ 8 22 133  4  6  2]
 [ 4  1  0 154  0  2]
 [ 9  4  2  2 129  0]
 [ 3  4  0  4  0 153]]
```

	precision	recall	f1-score	support
1	0.84	0.85	0.84	178
2	0.78	0.94	0.85	136
3	0.94	0.76	0.84	175
4	0.88	0.96	0.92	161
5	0.93	0.88	0.91	146
6	0.95	0.93	0.94	164
micro avg	0.88	0.88	0.88	960
macro avg	0.89	0.89	0.88	960
weighted avg	0.89	0.88	0.88	960

```
filename                                                    =
'final_models/ieee_SVC_kernal=sigmoid_k100_g05.sav'
```

2)

```
pipeline = Pipeline([('vect',
TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
('chi', SelectKBest(chi2, k=100)), ('clf',
SVC(kernel="sigmoid", gamma=5))])
```

Accuracy ----------->>>>> **93.22916666666666**

```
[[161  5  2  4  0  3]
 [ 1 152  9  1  0  3]
 [ 3  5 136  2  3  4]
 [ 2  2  0 147  0  2]
 [ 1  2  3  2 155  3]
 [ 0  0  1  2  0 144]]
```

	precision	recall	f1-score	support
1	0.96	0.92	0.94	175
2	0.92	0.92	0.92	166
3	0.90	0.89	0.89	153
4	0.93	0.96	0.95	153
5	0.98	0.93	0.96	166
6	0.91	0.98	0.94	147
micro avg	0.93	0.93	0.93	960
macro avg	0.93	0.93	0.93	960
weighted avg	0.93	0.93	0.93	960

```
filename =
'final_models/ieee_SVC_kernal=sigmoid_k100_g5.sav'
```

3)

```
pipeline = Pipeline([('vect',
TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
('chi', SelectKBest(chi2, k=100)), ('clf',
SVC(kernel="sigmoid", gamma=7))])
```

Accuracy ---------->>>>> **93.22916666666666**

```
[[155  3  2  3  3  3]
 [  4 145  2  3  0  1]
 [  2  6 133  1  3  0]
 [  2  2  0 163  1  3]
 [  2  1  6  1 154  5]
 [  4  2  0  0  0 145]]
```

	precision	recall	f1-score	support
1	0.92	0.92	0.92	169
2	0.91	0.94	0.92	155
3	0.93	0.92	0.92	145
4	0.95	0.95	0.95	171
5	0.96	0.91	0.93	169
6	0.92	0.96	0.94	151
micro avg	0.93	0.93	0.93	960
macro avg	0.93	0.93	0.93	960
weighted avg	0.93	0.93	0.93	960

```
filename =
'final_models/ieee_SVC_kernal=sigmoid_k100_g7.sav
```

SVC kernel = poly

1)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', SVC(kernel='poly', degree=3, C=1))])
```

```
14.583333333333334

 [[  0 176   0   0   0   0]
 [  0 140   0   0   0   0]
 [  0 157   0   0   0   0]
 [  0 155   0   0   0   0]
 [  0 169   0   0   0   0]
 [  0 163   0   0   0   0]]

              precision    recall  f1-score   support

           1       0.00      0.00      0.00       176
           2       0.15      1.00      0.25       140
           3       0.00      0.00      0.00       157
           4       0.00      0.00      0.00       155
           5       0.00      0.00      0.00       169
           6       0.00      0.00      0.00       163
   micro avg       0.15      0.15      0.15       960
   macro avg       0.02      0.17      0.04       960
weighted avg       0.02      0.15      0.04       960
```

```
filename = 'final_models/ieee_SVC_kernal=poly_k100_d3_c1.sav'
```

2)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', SVC(kernel='poly', degree=6, C=1))])
```

```
15.104166666666666

 [[  0   0   0 170   0   0]
 [  0   0   0 162   0   0]
 [  0   0   0 167   0   0]
 [  0   0   0 145   0   0]
 [  0   0   0 159   0   0]
 [  0   0   0 157   0   0]]

              precision    recall  f1-score   support

           1       0.00      0.00      0.00       170
           2       0.00      0.00      0.00       162
           3       0.00      0.00      0.00       167
           4       0.15      1.00      0.26       145
           5       0.00      0.00      0.00       159
           6       0.00      0.00      0.00       157
   micro avg       0.15      0.15      0.15       960
   macro avg       0.03      0.17      0.04       960
weighted avg       0.02      0.15      0.04       960
```

```
filename = 'final_models/ieee_SVC_kernal=poly_k100_d6_c1.sav'
```

LinearSVC

1)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', LinearSVC(C=1))])
```

91.77083333333333

```
[[119   2   2   5   2   7]
 [  1 159   2   1   3   1]
 [  7  13 140   1   2   2]
 [  2   0   0 141   0   2]
 [  3   1   3   0 162   3]
 [  6   2   0   3   3 160]]
```

	precision	recall	f1-score	support
1	0.86	0.87	0.87	137
2	0.90	0.95	0.92	167
3	0.95	0.85	0.90	165
4	0.93	0.97	0.95	145
5	0.94	0.94	0.94	172
6	0.91	0.92	0.92	174
micro avg	0.92	0.92	0.92	960
macro avg	0.92	0.92	0.92	960
weighted avg	0.92	0.92	0.92	960

```
filename = 'final_models/ieee_LinearSVC_k100_c1.sav'
```

2)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', LinearSVC(C=1))])
```

90.625

```
[[132   6   4   4   2   4]
 [  6 156   5   0   1   6]
 [  4  15 142   0   4   3]
 [  2   0   3 159   0   2]
 [  2   1   6   1 148   0]
 [  4   0   1   3   1 133]]
```

	precision	recall	f1-score	support
1	0.88	0.87	0.87	152
2	0.88	0.90	0.89	174
3	0.88	0.85	0.86	168
4	0.95	0.96	0.95	166
5	0.95	0.94	0.94	158
6	0.90	0.94	0.92	142
micro avg	0.91	0.91	0.91	960
macro avg	0.91	0.91	0.91	960
weighted avg	0.91	0.91	0.91	960

```
filename = 'final_models/ieee_LinearSVC_k1000_c1.sav'
```

3)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', LinearSVC(C=100))])
```

92.29166666666667

```
[[141   3   3   4   2   2]
 [  3 149   8   2   0   5]
 [  3   8 164   0   4   2]
 [  5   0   1 144   1   3]
 [  2   2   2   0 140   0]
 [  1   2   1   5   0 148]]
```

	precision	recall	f1-score	support
1	0.91	0.91	0.91	155
2	0.91	0.89	0.90	167
3	0.92	0.91	0.91	181
4	0.93	0.94	0.93	154
5	0.95	0.96	0.96	146
6	0.93	0.94	0.93	157
micro avg	0.92	0.92	0.92	960
macro avg	0.92	0.92	0.92	960
weighted avg	0.92	0.92	0.92	960

```
filename = 'final models/ieee LinearSVC k100 c100.sav'
```

4)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', LinearSVC(C=100))])
```

87.60416666666667

```
[[132   5   4   5   1   7]
 [  5 142  10   4   0   3]
 [ 10  19 120   1   3   6]
 [  3   2   1 149   0   4]
 [  3   1   8   2 143   1]
 [  3   4   1   3   0 155]]
```

	precision	recall	f1-score	support
1	0.85	0.86	0.85	154
2	0.82	0.87	0.84	164
3	0.83	0.75	0.79	159
4	0.91	0.94	0.92	159
5	0.97	0.91	0.94	158
6	0.88	0.93	0.91	166
micro avg	0.88	0.88	0.88	960
macro avg	0.88	0.88	0.88	960
weighted avg	0.88	0.88	0.88	960

```
filename = 'final models/ieee LinearSVC k1000 c100.sav
```

5)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', LinearSVC(C=10))])
```

89.58333333333334

```
[[131   1   3   5   0   4]
 [  4 143  12   1   1   5]
 [  8  12 138   2   5   1]
 [ 10   0   1 159   0   5]
 [  3   2   2   1 153   1]
 [  2   2   3   2   2 136]]
```

	precision	recall	f1-score	support
1	0.83	0.91	0.87	144
2	0.89	0.86	0.88	166
3	0.87	0.83	0.85	166
4	0.94	0.91	0.92	175
5	0.95	0.94	0.95	162
6	0.89	0.93	0.91	147
micro avg	0.90	0.90	0.90	960
macro avg	0.90	0.90	0.90	960
weighted avg	0.90	0.90	0.90	960

```
filename = 'final_models/ieee_LinearSVC_k1000_c10.sav'
```

6)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', LinearSVC(C=10))])
```

92.29166666666667

```
[[161   3   2   5   1   3]
 [  2 136   7   2   1   4]
 [  4  10 133   0   4   2]
 [  5   0   0 160   1   2]
 [  3   0   3   0 150   1]
 [  4   1   1   3   0 146]]
```

	precision	recall	f1-score	support
1	0.90	0.92	0.91	175
2	0.91	0.89	0.90	152
3	0.91	0.87	0.89	153
4	0.94	0.95	0.95	168
5	0.96	0.96	0.96	157
6	0.92	0.94	0.93	155
micro avg	0.92	0.92	0.92	960
macro avg	0.92	0.92	0.92	960
weighted avg	0.92	0.92	0.92	960

```
filename = 'final models/ieee LinearSVC k100 c10.sav'
```

KNN

1)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', KNeighborsClassifier(n_neighbors=3))])
```

```
79.375

[[119   5   3  10   7   7]
 [ 12 132   6   3   3   2]
 [ 17  18 117   3   5   3]
 [ 11   5   1 148   3   2]
 [ 12   9   8   2 126   4]
 [ 16   8   1  12   0 120]]

              precision    recall  f1-score   support

           1       0.64      0.79      0.70       151
           2       0.75      0.84      0.79       158
           3       0.86      0.72      0.78       163
           4       0.83      0.87      0.85       170
           5       0.88      0.78      0.83       161
           6       0.87      0.76      0.81       157
   micro avg       0.79      0.79      0.79       960
   macro avg       0.80      0.79      0.79       960
weighted avg       0.81      0.79      0.80       960
```

```
filename = 'final_models/ieee_knn_k100_n3.sav'
```

2)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', KNeighborsClassifier(n_neighbors=7))])
```

```
85.52083333333333

[[138   4   4  14   3   5]
 [  5 145   2   2   0   4]
 [ 10   8 121   2   5   1]
 [  8   5   2 142   0   3]
 [ 10   8   6   3 138   4]
 [  6   4   3   6   2 137]]
              precision    recall  f1-score   support

           1       0.78      0.82      0.80       168
           2       0.83      0.92      0.87       158
           3       0.88      0.82      0.85       147
           4       0.84      0.89      0.86       160
           5       0.93      0.82      0.87       169
           6       0.89      0.87      0.88       158
   micro avg       0.86      0.86      0.86       960
   macro avg       0.86      0.86      0.86       960
weighted avg       0.86      0.86      0.86       960
```

```
filename = 'final_models/ieee_knn_k100_n7.sav'
```

3)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', KNeighborsClassifier(n_neighbors=3))])
```

61.458333333333336

```
[[102  12   9   3   3   8]
 [ 38 110  14   2   2   4]
 [ 24  20  85   1   5   4]
 [ 36  28  13  89   0   7]
 [ 29  16  20   7 114   1]
 [ 29  22   9   4   0  90]]
```

	precision	recall	f1-score	support
1	0.40	0.74	0.52	137
2	0.53	0.65	0.58	170
3	0.57	0.61	0.59	139
4	0.84	0.51	0.64	173
5	0.92	0.61	0.73	187
6	0.79	0.58	0.67	154
micro avg	0.61	0.61	0.61	960
macro avg	0.67	0.62	0.62	960
weighted avg	0.69	0.61	0.63	960

```
filename = 'final models/ieee knn k1000 n3.sav'
```

4)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', KNeighborsClassifier(n_neighbors=7))])
```

60.520833333333336

```
[[102  32   2   2   3   5]
 [ 10 125   4   0   0   1]
 [ 17  45  88   0   8   6]
 [ 24  75   1  68   0   7]
 [ 17  53   5   2  87   1]
 [  9  43   4   3   0 111]]
```

	precision	recall	f1-score	support
1	0.57	0.70	0.63	146
2	0.34	0.89	0.49	140
3	0.85	0.54	0.66	164
4	0.91	0.39	0.54	175
5	0.89	0.53	0.66	165
6	0.85	0.65	0.74	170
micro avg	0.61	0.61	0.61	960
macro avg	0.73	0.62	0.62	960
weighted avg	0.75	0.61	0.62	960

```
filename = 'final models/ieee knn k1000 n7.sav'
```

Random Forest

1)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf',RandomForestClassifier(n_estimators=200, max_depth=3,
random_state=0))])
```

```
90.3125

[[137   2   5   7   1   8]
 [  3 152   5   8   1  11]
 [  5   7 134   2   3   4]
 [  1   0   0 162   0   2]
 [  6   0   6   2 142   1]
 [  1   0   0   2   0 140]]

              precision    recall  f1-score   support

           1       0.90      0.86      0.88       160
           2       0.94      0.84      0.89       180
           3       0.89      0.86      0.88       155
           4       0.89      0.98      0.93       165
           5       0.97      0.90      0.93       157
           6       0.84      0.98      0.91       143
   micro avg       0.90      0.90      0.90       960
   macro avg       0.90      0.91      0.90       960
weighted avg       0.91      0.90      0.90       960
```

```
filename = 'final_models/ieee_randomf_k1000_ne200.sav'
```

2)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf',RandomForestClassifier(n_estimators=100, max_depth=3,
random_state=0))])
```

```
90.10416666666666

[[130   2   2   7   3   8]
 [  6 131  12   1   1   5]
 [  2   6 138   2   4   6]
 [  1   0   1 156   0   3]
 [  8   1   5   4 151   1]
 [  1   0   0   3   0 159]]

              precision    recall  f1-score   support

           1       0.88      0.86      0.87       152
           2       0.94      0.84      0.89       156
           3       0.87      0.87      0.87       158
           4       0.90      0.97      0.93       161
           5       0.95      0.89      0.92       170
           6       0.87      0.98      0.92       163
   micro avg       0.90      0.90      0.90       960
   macro avg       0.90      0.90      0.90       960
weighted avg       0.90      0.90      0.90       960
```

```
filename = 'final_models/ieee_randomf_k1000_ne100.sav'
```

3)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf',RandomForestClassifier(n_estimators=100, max_depth=3,
random_state=0))])
```

```
90.72916666666667
[[126   6   0   4   3  10]
 [  7 116   8   5   0   4]
 [  9   3 147   3   6   4]
 [  0   2   0 169   0   4]
 [  2   0   3   2 153   2]
 [  0   1   0   0   1 160]]
              precision    recall  f1-score   support

           1       0.88      0.85      0.86       149
           2       0.91      0.83      0.87       140
           3       0.93      0.85      0.89       172
           4       0.92      0.97      0.94       175
           5       0.94      0.94      0.94       162
           6       0.87      0.99      0.92       162

   micro avg       0.91      0.91      0.91       960
   macro avg       0.91      0.90      0.90       960
weighted avg       0.91      0.91      0.91       960
```

```
filename = 'final_models/ieee_randomf_k100_ne100.sav'
```

Naïve Bayes

Multinomial

1)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', MultinomialNB())])
```

```
84.89583333333334
[[132   2   3   8   3   7]
 [  6 136   8   5   1   3]
 [ 14  13 129   1   7   5]
 [  5   1   2 153   0   7]
 [  9   3  11   2 122   2]
 [  5   7   2   3   0 143]]
              precision    recall  f1-score   support

           1       0.77      0.85      0.81       155
           2       0.84      0.86      0.85       159
           3       0.83      0.76      0.80       169
           4       0.89      0.91      0.90       168
           5       0.92      0.82      0.87       149
           6       0.86      0.89      0.87       160

   micro avg       0.85      0.85      0.85       960
   macro avg       0.85      0.85      0.85       960
weighted avg       0.85      0.85      0.85       960
```

```
filename = 'final_models/ieee_nb_multi_k100.sav'
```

2)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', MultinomialNB())])
```

```
78.125
[[113  11   9  16   5  12]
 [ 10 109  13  10   0   4]
 [ 10  17 113   3   7   8]
 [  5   3   0 152   1  10]
 [  3   4  10   2 138   7]
 [ 12   5   5   8   0 125]]
              precision    recall  f1-score   support

           1       0.74      0.68      0.71       166
           2       0.73      0.75      0.74       146
           3       0.75      0.72      0.73       158
           4       0.80      0.89      0.84       171
           5       0.91      0.84      0.88       164
           6       0.75      0.81      0.78       155

   micro avg       0.78      0.78      0.78       960
   macro avg       0.78      0.78      0.78       960
weighted avg       0.78      0.78      0.78       960
```

```
filename = 'final_models/ieee_nb_multi_k1000.sav'
```

Bernuli

3)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None,
fit_prior=True))])
```

```
90.52083333333333
[[124   4   1  11   2   9]
 [  1 164   5   1   1   2]
 [ 10  15 138   1   3   2]
 [  4   2   0 152   0   1]
 [  6   0   1   2 147   0]
 [  0   2   1   3   1 144]]
              precision    recall  f1-score   support

           1       0.86      0.82      0.84       151
           2       0.88      0.94      0.91       174
           3       0.95      0.82      0.88       169
           4       0.89      0.96      0.92       159
           5       0.95      0.94      0.95       156
           6       0.91      0.95      0.93       151

   micro avg       0.91      0.91      0.91       960
   macro avg       0.91      0.91      0.90       960
weighted avg       0.91      0.91      0.90       960
```

```
filename = 'final_models/ieee_nb_bernuli_k100.sav'
```

4)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None,
fit_prior=True))])
```

```
84.27083333333333
[[131  10   5   8   1   9]
 [  5 124  16   6   2   3]
 [  8  11 110   2   2   2]
 [  6   1   2 159   1   5]
 [ 11   2   7   2 142   1]
 [  6   6   3   8   0 143]]
              precision    recall  f1-score   support

           1       0.78      0.80      0.79       164
           2       0.81      0.79      0.80       156
           3       0.77      0.81      0.79       135
           4       0.86      0.91      0.89       174
           5       0.96      0.86      0.91       165
           6       0.88      0.86      0.87       166

   micro avg       0.84      0.84      0.84       960
   macro avg       0.84      0.84      0.84       960
weighted avg       0.85      0.84      0.84       960
```

```
filename = 'final_models/ieee_nb_bernuli_k1000.sav'
```

Logistic Regression

1)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', LogisticRegression(random_state=0))])
```

```
91.45833333333333
[[144   2   4   7   1   4]
 [  5 138   1   1   0   3]
 [  7   8 141   0   1   1]
 [  2   0   0 174   0   3]
 [  2   3   5   4 128   3]
 [  3   4   1   7   0 153]]
              precision    recall  f1-score   support

           1       0.88      0.89      0.89       162
           2       0.89      0.93      0.91       148
           3       0.93      0.89      0.91       158
           4       0.90      0.97      0.94       179
           5       0.98      0.88      0.93       145
           6       0.92      0.91      0.91       168

   micro avg       0.91      0.91      0.91       960
   macro avg       0.92      0.91      0.91       960
weighted avg       0.92      0.91      0.91       960
```

```
filename = 'final_models/ieee_lr_k100.sav'
```

2)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=1000)),
                     ('clf', LogisticRegression(random_state=0))])
```

```
89.89583333333333
[[145   4   1  15   1   6]
 [  3 146   3   2   1   1]
 [  4   6 129   1   7   3]
 [  2   0   0 145   0   3]
 [  5   1   5   1 146   5]
 [  7   4   2   4   0 152]]
              precision    recall  f1-score   support

           1       0.87      0.84      0.86       172
           2       0.91      0.94      0.92       156
           3       0.92      0.86      0.89       150
           4       0.86      0.97      0.91       150
           5       0.94      0.90      0.92       163
           6       0.89      0.90      0.90       169

   micro avg       0.90      0.90      0.90       960
   macro avg       0.90      0.90      0.90       960
weighted avg       0.90      0.90      0.90       960
```

```
filename = 'final_models/ieee_lr_k1000.sav'
```

Decision Trees

1)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', DecisionTreeClassifier(max_depth=2))])
```

```
46.875
[[  0   0 153   7   6   0]
 [  0   0 167   2   1   0]
 [  0   0 139   0   4   0]
 [  0   0   5 150   1   0]
 [  0   0   4   0 161   0]
 [  0   0 152   8   0   0]]

              precision    recall  f1-score   support

           1       0.00      0.00      0.00       166
           2       0.00      0.00      0.00       170
           3       0.22      0.97      0.36       143
           4       0.90      0.96      0.93       156
           5       0.93      0.98      0.95       165
           6       0.00      0.00      0.00       160

   micro avg       0.47      0.47      0.47       960
   macro avg       0.34      0.48      0.37       960
weighted avg       0.34      0.47      0.37       960
```

```
filename = 'final_models/ieee_dt_k100.sav'
```

2)

```
pipeline = Pipeline([('vect', TfidfVectorizer(ngram_range=(1, 1), sublinear_tf=True)),
                     ('chi', SelectKBest(chi2, k=100)),
                     ('clf', DecisionTreeClassifier(max_depth=7))])
```

```
89.16666666666667
[[128   7   0   6   3   4]
 [  3 131   7   3   1   3]
 [  6  13 152   3   1   4]
 [  3   5   1 143   0   4]
 [  3   4   4   0 156   0]
 [ 12   0   1   1   2 146]]
              precision    recall  f1-score   support

           1       0.83      0.86      0.84       148
           2       0.82      0.89      0.85       148
           3       0.92      0.85      0.88       179
           4       0.92      0.92      0.92       156
           5       0.96      0.93      0.95       167
           6       0.91      0.90      0.90       162

   micro avg       0.89      0.89      0.89       960
   macro avg       0.89      0.89      0.89       960
weighted avg       0.89      0.89      0.89       960
```

```
filename = 'final_models/ieee_dt_k100_md7.sav'
```

From the all classifiers used to find the best one to doing profiling, SVC is being selected with sigmoid kernel with parameters values k = 100, gamma = 7 and has the highest accuracy of 93.22916666666666.

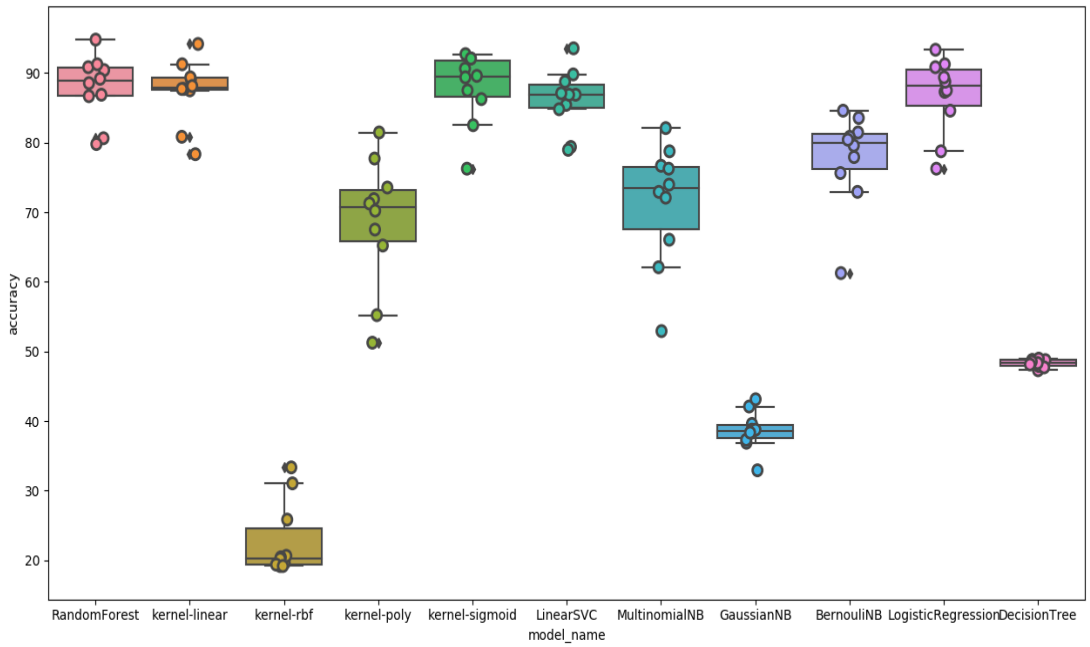


Figure 7.1 – Plot chart of accuracies of models

After best classifier is chosen, some profiles are being created using that model with different methods for lecturers of the faculty of Information Technology and results are showed under the Appendix B in this report. To evaluate those results, the user interests are being collected manually and compare those results with the system generated values.

Some of the results are being collected manually from the lectures and their results generated through the system as follows.

- ✓ Dr. Thushari Silva

Manually collected results

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
Medium	High	Medium	Low	Low	Medium

- ✓ System generated profile for Dr. Thushari Silva based on different methods

As sum of each array length into % [probability = True](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
20.0	30.0	30.0	10.0	10.0	0.0

As length of all abstracts into % [probability = True](output sum != 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
4.35	4.35	13.04	4.35	0.0	0.0

As length of all abstracts into % [probability = False](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
21.74	65.22	4.35	4.35	4.35	12.0

Using LSA (output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
13.04	43.47	0.0	21.73	17.39	0.0

- ✓ Dr. Upeksha Ganegoda

Manually collected results

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
Medium	Low	Low	No	High	Medium

- ✓ System generated profile for Dr. Upeksha Ganegoda based on different methods

As sum of each array length into % [probability = True](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
7.69	7.69	30.77	0.0	53.85	0.0

As length of all abstracts into % [probability = True](output sum != 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
0.0	4.55	27.27	0.0	4.55	0.0

As length of all abstracts into % [probability = False](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
18.18	40.91	22.73	0.0	18.18	0.0

Using LSA (output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
13.63	31.81	0.0	27.27	18.18	0.0

- ✓ Professor Asoka Karunananda

Manually collected results

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
High	Low	Low	No	No	High

- ✓ System generated profile for Professor Asoka Karunananda based on different methods

As sum of each array length into % [probability = True](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
53.85	15.38	3.85	5.77	0.0	21.15

As length of all abstracts into % [probability = True](output sum != 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
28.0	8.0	2.0	3.0	0.0	11.0

As length of all abstracts into % [probability = False](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
41.0	39.0	2.0	6.0	0.0	12.0

Using LSA (output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
20.0	5.0	0.0	56.99	8.0	0.0

After evaluate the results, the first method was chosen is suitable for generating the user profiles and the system has developed based on that method.

7.3 Journal Recommendation module for manuscript

In this module is released few results. User can submit their manuscript paper and it should be IEEE research paper format. Next extract necessary details (title,abstract,keywords) from manuscript. In step has included pre-processing part and it will remove unnecessary data. I used LDA to SVM model to get best theme for the submit paper. Another part is generating journals profiling part. It is scrapped data from source and train a model based on LDA model. Next profile module and journal

module are compared using matching algorithm.Finaly release few results. Those are Subject are for the manuscript, Top suggesting journals, exploration of productivity factors.

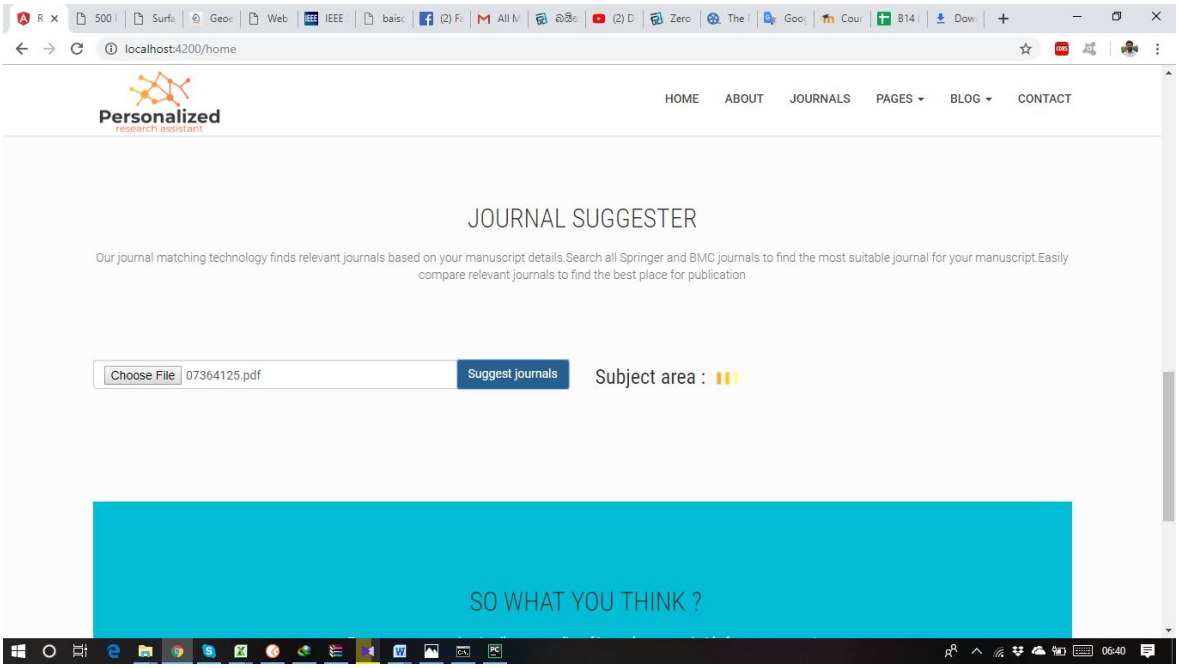


Figure 7.2: uploading pdf

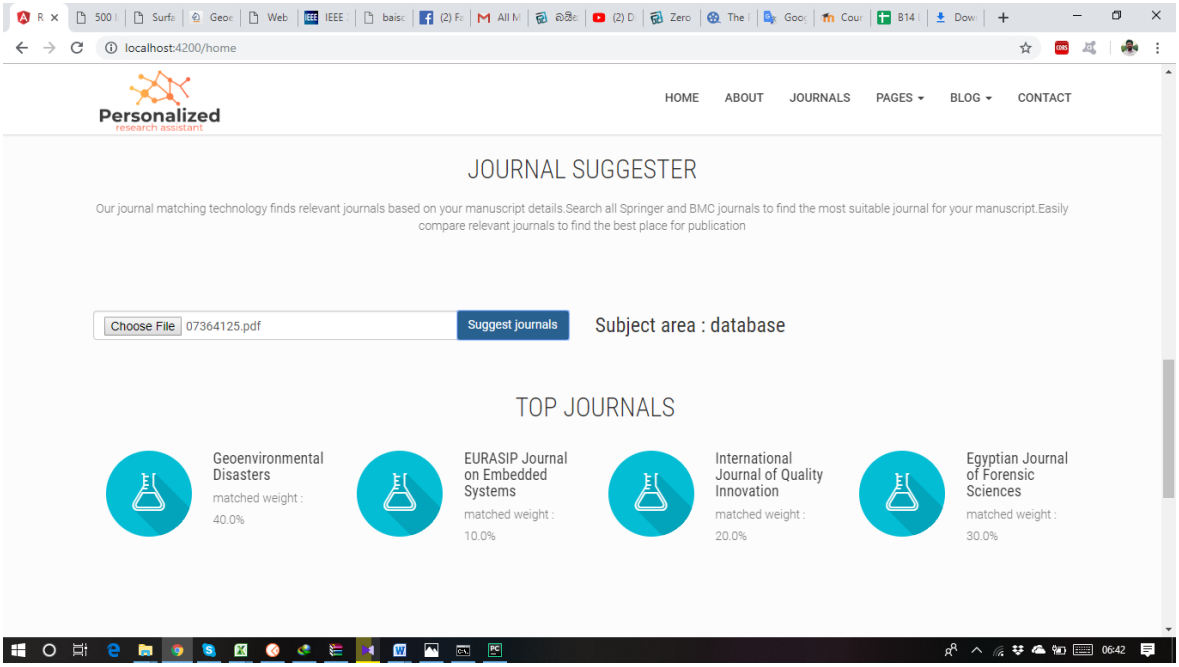


Figure 7.3: Top suitable journals with matched weight

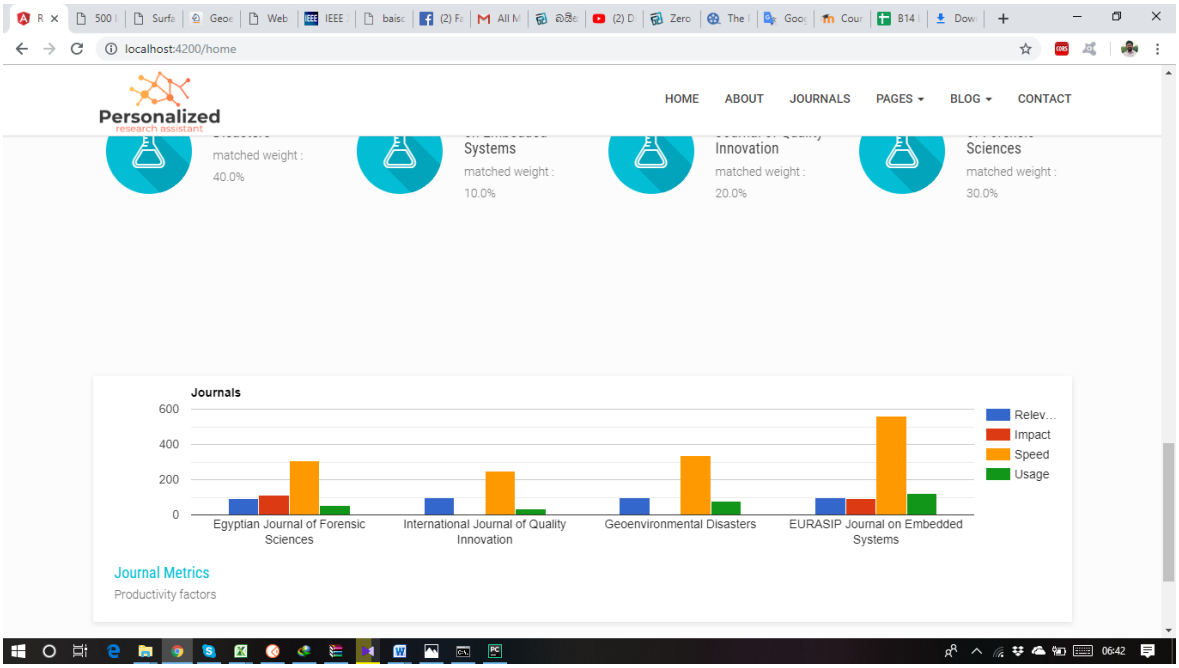


Figure 7.4: Exploration of productivity factors

7.4 The connectivity detection module

The proposed solution is provided suggestions of collaborators to authors from multiple networks. The approach is giving positive results. It is working with the Google scholar data which have some drawback. The major drawback is the identification of different names that have the same author in different research paper publications. Therefore need to be having a pre-processing step to reduce the false positive results that are provided.

The number of research papers that have published is varied from each author. Some Authors have more than 500 publications. Here, this approach uses data of the 10 research paper title of author and 3 research paper title of co-authors. With the purpose of testing the accuracy of the result gathered all data of the author thushari silva. But the result that generates with those two types of data sets are somewhat different. It implies that the level of accuracy has an impact from the number of research paper. But to handle a huge amount of data need high processing power.

Below pair of graphs shows that the difference between the result with the two type of datasets. Graphs show the details of the same author named Thushari Silva. The actual count of the research papers that have published by a particular author is 22 according to the details that obtain from Google Scholar. It means the first graph shows the actual result of the particular author and second graph shows the result of limited research papers.

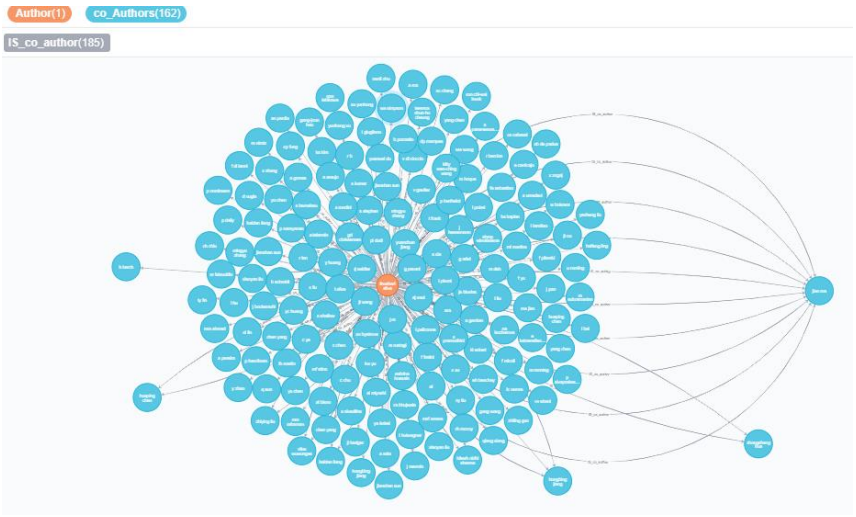


Figure 7.5: The co-authorship network of the one author with the limit of 25 research papers

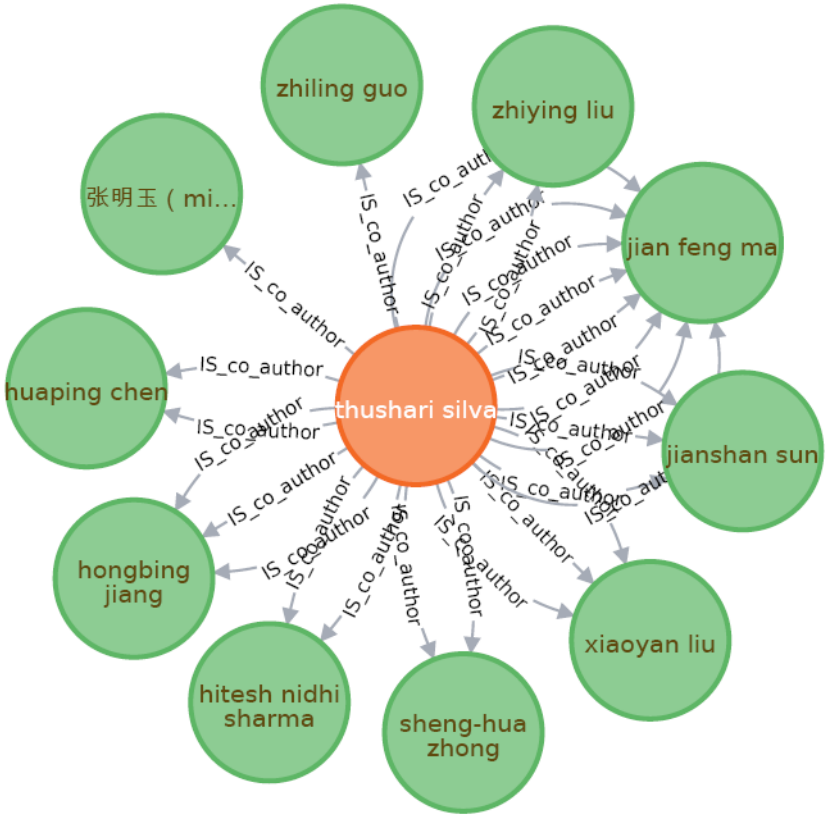


Figure 7.6: The co-authorship network of the one author with the limit of ten research papers

Here is segment of the result of co-authors relationship count which have all co-authors in data file. This is related to the first graph.

"Author"	"rel_count"
{"Author_name":"thushari silva"}	185
{"co_authors":" jian ma "}	10
{"co_authors":" chen yang "}	3
{"co_authors":" jianshan sun "}	3
{"co_authors":" hongbing jiang "}	3
{"co_authors":" xiaoyan liu "}	2
{"co_authors":" vilas wuwongse "}	2
{"co_authors":" huaping chen"}	2
{"co_authors":"chen yang "}	2
{"co_authors":" zhongsheng hua"}	2
{"co_authors":" haidan liang"}	2
{"co_authors":" zhiying liu "}	2
{"co_authors":" hitesh nidhi sharma"}	2

Figure 7.7: Table containing relationship count of co-authors with all research paper titles

Here is the result of co-authors relationship count which have limited co-authors in the data file. This is related to the second graph.

"co_author"	"rel_count"
"jian ma"	8
"xiaoyan liu"	3
"jianshan sun"	3
"hongbing jiang"	3
"sheng-hua zhong"	2
"zhiying liu"	2
"huaping chen"	2
"hitesh nidhi sharma"	2
"mingyu zhang"	1
"zhiling guo"	1

Figure 7.8: Table containing relationship counts of co-authors with 10 research paper titles

Above two tables show that the difference of the prediction. Both tables provide Jian ma as a strongest connected co-author of the author Thushari Silva. The first table shows count as 10 and the second one is shown as 8. But in the first table, the second strongest connected author is Chen yang and the second table is Xiaoyan Liu. However, the actual result should be Chen Yang. It implies that if have all details of the author's publication cause to increase the level of accuracy.

7.5 Collaborators Recommendation module

In this module, main objective is to recommend the users who are most related and which have the highest connectivity to the newly logged user into the system. The previously decided approach was different and that was to be changed because of some drawbacks.

First the Cosine similarity algorithm was used and after identifying the Kendall's tau algorithm the drawbacks were seem to be reduced so that the Kendall's tau algorithm was used.

Kendall's tau algorithm can be used for a large set of data. Since the Recommendation part gets a limited value of data related to the outputs of the two sub modules it was seemed to be a bit challenge using it.

```
import csv
import math
from heapq import nlargest
import scipy.stats as stats

users = []
logged_user = [62, 29, 9]
ranks = {}

with open('datasets/testdatarelevance.csv') as file:
    read_file = csv.reader(file, delimiter=',')

    for row in read_file:
        users.append(row)

    print(users)

users_tau = {}
users_p = {}
pearson_cor = {}
kandals_values = {}
for count in range(1, users.__len__()):
    user_details = []
    u_name = users[count][0]
    user_details.append(float(users[count][1]))
    user_details.append(float(users[count][2]))
    user_details.append(float(users[count][3]))

    ar_c = []
    ar_d = []

    # kandals tau
    if(user_details[0] > user_details[1] and user_details[0] > user_details[2]):
```

Figure 7.9: Kendall's tau testing (1)

```
    # kandals tau
    if(user_details[0] > user_details[1] and user_details[0] > user_details[2]):
        c = 2
        d =0
        ar_c.append(c)
        ar_d.append(d)
    elif(user_details[0] > user_details[1] and user_details[0] < user_details[2]):
        c = 1
        d = 1
        ar_c.append(c)
        ar_d.append(d)
    else:
        c = 0
        d = 2
        ar_c.append(c)
        ar_d.append(d)

    if (user_details[1] > user_details[2]):
        c = 1
        d = 0
        ar_c.append(c)
        ar_d.append(d)
    else:
        c = 0
        d = 1
        ar_c.append(c)
        ar_d.append(d)

    c_sum = 0
    for i in ar_c:
        c_sum = c_sum + i
```

Figure 7.10: Kendall's tau testing (2)

Because of the drawbacks of the Kendall's tau algorithm a new algorithm should be found for implementation. Then the Pearson's correlation algorithm which is a bit similar to the Cosine similarity algorithm was found.

```
def cal_pearson_correlation(new_user):

    log_user = []
    # create an array with logged user values
    for i in range(2, 8):
        log_user.append(new_user[i])

    users = get_all_users()

    correlation_values = {}
    u_id_with_name = {}

    for u in range(1, users.__len__()):
        user = users[u]
        related_user = []
        u_id = user[0]
        u_name = user[1]
        for index in range(2, 8):
            related_user.append(int(user[index]))

        value = stats.stats.pearsonr(log_user, related_user)

        correlation_values[u_name] = (value[0] * 10)
        u_id_with_name[u_name] = u_id
    print(u_id_with_name)

    sorted_set = nlargest(10, correlation_values, key=correlation_values.get)
    for user in sorted_set:
        print(user + ' : ' + str(correlation_values.get(user)))
```

Figure 7.11: Pearson's correlation testing

Pearson's Correlation Algorithm

New User Values - 10, 6, 13, 50, 15, 16
Need of users - 3
Cut off marks
Connectivity - 7
Relevance - 7

{'User 1': '0001', 'User 2': '0002', 'User 3': '0003', 'User 4': '0004', 'User 5': '0005', 'User 6': '0006', 'User 7': '0007', 'User 8': '0008', 'User 9': '0009', 'User 10': '0010', 'User 11': '0011', 'User 12': '0012', 'User 13': '0013', 'User 14': '0014', 'User 15': '0015', 'User 16': '0016', 'User 17': '0017', 'User 18': '0018', 'User 19': '0019', 'User 20': '0020', 'User 21': '0021', 'User 22': '0022', 'User 23': '0023', 'User 24': '0024', 'User 25': '0025', 'User 26': '0026', 'User 27': '0027', 'User 28': '0028', 'User 29': '0029', 'User 30': '0030', 'User 31': '0031'}

User 10 : 10.0

User 19 : 9.7409483933131

User 20 : 8.879189259715938

User 17 : 8.574119803377076

User 26 : 8.438079697253777

User 30 : 8.406571348924034

User 27 : 5.437874215115995

User 3 : 0.09620869158223383

User 22 : -0.1207718275664218

User 16 : -0.15426789015466927

Users who have relevance value more than 7:

['User 10', 'User 19', 'User 20', 'User 17', 'User 26', 'User 30']

Users who have connectivity value more than 7:

[['User 2', '8'], ['User 6', '7'], ['User 10', '9'], ['User 18', '9'], ['User 24', '7], ['User 26', '8']]

FuzzyWuzzy results:

[('0010', 75)]

[('0010', 75)]

[('0010', 100)]

[('0010', 75)]

[('0020', 75)]

[('0026', 100)]

Common set:

['0010', '0026']

[['User 10'], ['User 26']]

Figure 7.12: Pearson's correlation testing data

There were some drawbacks be founded as recommending the nonrelated users who have a faraway distant regarding the values of the new user. So that new algorithm should be found. As a result of that Spearman's correlation algorithm was use because the outputs received were much more realistic than the Pearson's algorithm.

Spearman's Correlation Algorithm

New User (ML, BD, DM, CV, Bio, AI)	Top 10 users	Top 10 users Values	Spearman's correlation of top 10 users
10, 6, 13, 50, 15, 16	User 10	10, 6, 13, 50, 15, 16	10.0
	User 17	10, 0, 2, 41, 21, 26	-0.685714285714285
	User 26	6, 17, 3, 39, 14, 21	-0.9714285714285715
	User 20	10, 7, 13, 30, 18, 22	-2.742857142857142
	User 30	20, 15, 5, 50, 2, 8	-3.6571428571428566
	User 19	5, 6, 0, 69, 7, 13	-7.942857142857143
	User 27	1, 9, 15, 32, 2, 41	-24.628571428571426
	User 3	10, 36, 4, 20, 5, 25	-48.91428571428571
	User 18	30, 1, 39, 15, 6, 9	-60.17142857142858
	User 16	0, 18, 46, 16, 12, 8	-63.2

Common set of users	Relevance	Connectivity	Value
User 10	10.0	9	19.0
User 26	-0.9714285714285715	8	7.0285714285714285
User 20	-2.742857142857142	5	2.257142857142858
User 18	-60.17142857142858	9	-51.17142857142858
User 16	-63.2	6	-57.2

Figure 7.13: Spearman's correlation output data

Spearman vs. Pearson

New User Values - 10, 6, 13, 50, 15, 16

Spearman's Correlation Algorithm top 10	Values	Pearson's Correlation Algorithm top 10	Values
User 10	10, 6, 13, 50, 15, 16	User 10	10, 6, 13, 50, 15, 16
User 17	10, 0, 2, 41, 21, 26	User 19	5, 6, 0, 69, 7, 13
User 26	6, 17, 3, 39, 14, 21	User 20	10, 7, 13, 30, 18, 22
User 20	10, 7, 13, 30, 18, 22	User 17	10, 0, 2, 41, 21, 26
User 30	20, 15, 5, 50, 2, 8	User 26	6, 17, 3, 39, 14, 21
User 19	5, 6, 0, 69, 7, 13	User 30	20, 15, 5, 50, 2, 8
User 27	1, 9, 15, 32, 2, 41	User 27	1, 9, 15, 32, 2, 41
User 3	10, 36, 4, 20, 5, 25	User 3	10, 36, 4, 20, 5, 25
User 18	30, 1, 39, 15, 6, 9	User 22	0, 4, 6, 8, 22, 50
User 16	0, 18, 46, 16, 12, 8	User 16	0, 18, 46, 16, 12, 8

Figure 7.14: Comparison between Spearman's algorithm and Pearson's algorithm.

Before the two algorithms are compared, a manual set of an expected output was prepared manually for testing purpose. The Spearman's algorithm provided the best matches for that manually created dataset so it was chosen for implementation.

When using a hybrid approach for implementation, finding the best string matching algorithm was difficult. I tried to include several content based approaches such as TF/IDF and Jacquard similarity method. But finally the Fuzzy String Matching algorithm was the best to use compared to our approach.

When it comes to the evaluation finding the relevant techniques is not an easy task because experimental efforts are mandatory here.

7.6 Summary

The chapter seven is described the evaluation and result comparing different results under the different conditions. It is provided detailed evaluate about the the solution to see whether objectives have been achieved.

Chapter 08

Conclusion

8.1 Introduction

Previous chapter discussed about the evaluation of the system and this chapter will discuss about the overall achievement of the system, how the problem is being encountered with the challenges faced while the successful implementation of the project.

For the ease of development and research of the project, it is being divided into four main modules as profiling, research opportunities recommendation module, journal recommendation module and supervisor recommendation module. These four modules are being conducted presently with the intention of developing a successful product. The main objective of the product is to develop a single platform for the researchers based on the user profile to gather all the services they search from several sites. Information was extracted from the popular sites used by the researchers such as Google Scholar and IEEE. When developing the product, finding the relevant technologies for each module is bit difficult. First in the system create a profile automatically for each researcher by using the abstracts from Research Gate profile of the user. In here the Google scholar name of the user is being asked when logging into the system considering about building up a quality and reliable profile with the correct information about the user. If the researcher needs somewhere to publish them recommend platform for that. The system reads the content of the research paper and suggests the best place to publish that. Other than that the social connection network is being created for the ease of the user. Recommending collaborators is the other part of this system. This is done by using the outputs given by the profiling module and social connection module There is no a single site to handle all those mentioned sub modules. Therefore the system benefits the users to get all the services from a single platform by saving, time and serving better options.

Chapter 09

References

[1] Hong, K., Jeon, H., Jeon, C., “Personalized Research Paper Recommendation System using Keyword Extraction Based on UserProfile”, Department of Computer Science & Engineering, Hanyang University, Korea, Agency for Defense Development, Seoul, Korea.

[2] "ScholarMate." Wikipedia, The Free Encyclopedia, 9 May. 2018. Web. 20 Sep. 2018.

[3] “Jane.biosemantics.org. (2018). *Journal / Author Name Estimator*.” [online] Available at: http://jane.biosemantics.org/ [Accessed 20 Sep. 2018].

[4] “www.springer.com. (2018). *Springer journal suggester*.” [online] Available at: https://www.springer.com/gp/authors-editors/journal-author/journal-author-helpdesk/selecting-a-journal/1258 [Accessed 20 Sep. 2018].

[5] Hong, K., Jeon, H., Jeon, C., “Personalized Research Paper Recommendation System using Keyword Extraction Based on UserProfile”, Department of Computer Science & Engineering, Hanyang University, Korea, Agency for Defense Development, Seoul, Korea.

[6] Personalization of Web Search Results Based on User Profiling July 2008 DOI: 10.1109/ICETET.2008.28 Conference: ICETET-2008 Snigdha Gupta, Saral Jain, Mohammad Kaz

[7] Thushari Silva, Jian Ma, (2017) "Expert profiling for collaborative innovation: big data perspective", Information Discovery and Delivery, Vol. 45 Issue: 4, pp.169-180,

[8] S. M. Billah and S. Gauch, "Social network analysis for predicting emerging researchers," *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, Lisbon, 2015, pp. 27-35.

[9] Shun-Hua Tan, Miao Chen and Guo-Hai Yang, "User behavior mining on large scale web log data," *The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding*, Chengdu, 2010, pp. 60-63.

[10] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal and J. Han, "Co-author Relationship Prediction in Heterogeneous Bibliographic Networks," *2011 International Conference on Advances in Social Networks Analysis and Mining*, Kaohsiung, 2011, pp. 121-128.

[11] D. N. Nakornpanom and S. Prom-on, "Identification of relevant experts using academic social network mining," *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, Nagoya, 2017, pp. 711-714.

[12] B. Qiu, K. Ivanova, J. Yen and P. Liu, "Behavior Evolution and Event-Driven Growth Dynamics in Social Networks," *2010 IEEE Second International Conference on Social Computing*, Minneapolis, MN, 2010, pp. 217-224.

[13] M. Alsukhni and Y. Zhu, "Interactive visualization of the social network of research collaborations," *2012 IEEE 13th International Conference on Information Reuse & Integration (IRI)*, Las Vegas, NV, 2012, pp. 247-254.
[14] V. Martha, W. Zhao and X. Xu, "A study on Twitter user-follower network a network based analysis," *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, Niagara Falls, ON, 2013, pp. 1405-1409.

[15] S. L. Toral, N. Bessis, M. R. Martinez-Torres, F. Franc, F. Barrero and F. Xhafa, "An Exploratory Social Network Analysis of Academic Research Networks," *2011 Third International Conference on Intelligent Networking and Collaborative Systems*, Fukuoka, 2011, pp. 21-26.

[16] A. Bartal, E. Sasson and G. Ravid, "Predicting Links in Social Networks Using Text Mining and SNA," *2009 International Conference on Advances in Social Network Analysis and Mining*, Athens, 2009, pp. 131-136.

[17] Y. Dong, S. Liu and J. Chai, "Research of hybrid collaborative filtering algorithm based on news recommendation," *2016 9th International Congress on Image and Signal Processing, Bio Medical Engineering and Informatics (CISP-BMEI)*, Datong, 2016, pp. 898-902.

[18] D. Pathak, S. Matharia and C. N. S. Murthy, "ORBIT: Hybrid movie recommendation engine," *2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN)*, Tirunelveli, 2013, pp. 19-24.

[19] B. Xiang, Z. Zhang, H. Dong, Q. Wu and L. Hu, "Research of mobile recommendation system based on hybrid recommendation technology," *2013 3rd International Conference on Consumer Electronics, Communications and Networks*, Xianning, 2013, pp. 508-512.

[20] X. Zhu, H. Ye and S. Gong, "A personalized recommendation system combining case-based reasoning and user-based collaborative filtering," *2009 Chinese Control and Decision Conference*, Guilin, 2009, pp. 4026-4028.

[21] A. S. Tewari, A. Kumar and A. G. Barman, "Book recommendation system based on combine features of content based filtering, collaborative filtering and association rule mining," *2014 IEEE International Advance Computing Conference (IACC)*, Gurgaon, 2014, pp. 500-503.

[22] Q. Zhu, M. Shyu and H. Wang, "VideoTopic: Content-Based Video Recommendation Using a Topic Model," *2013 IEEE International Symposium on Multimedia*, Anaheim, CA, 2013, pp. 219-222.

[23] J. Sun *et al.*, "A Novel Approach for Personalized Article Recommendation in Online Scientific Communities," *2013 46th Hawaii International Conference on System Sciences*, Wailea, Maui, HI, 2013, pp.

[24] X. Cai *et al.*, "Learning Collaborative Filtering and Its Application to People to People Recommendation in Social Networks," *2010 IEEE International Conference on Data Mining*, Sydney, NSW, 2010, pp. 743-748.

[25] S. Manju and M. Thenmozhi, "Privacy Preserving Collaborative Filtering Approach for Recommendation System," *2018 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, 2018, pp. 1-5.

[26] L. Toral, N. Bessis, M. R. Martinez-Torres, F. Franc, F. Barrero and F. Xhafa, "An Exploratory Social Network Analysis of Academic Research Networks," *2011 Third International Conference on Intelligent Networking and Collaborative Systems*, Fukuoka, 2011, pp. 21-26.

[27] S. M. Billah and S. Gauch, "Social network analysis for predicting emerging researchers," *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, Lisbon, 2015, pp. 27-35.

[28] A. Bartal, E. Sasson and G. Ravid, "Predicting Links in Social Networks Using Text Mining and SNA," *2009 International Conference on Advances in Social Network Analysis and Mining*, Athens, 2009, pp. 131-136.

Appendix A

Individual Contribution to the Project

Name: Rathnayake K.V.R.A.M.

In a profile-based system, the most important part is building up the profile according to the considered circumstances of the project. All the other modules in this system use the output which is given from the profile and because of that giving the relevant and accurate information about the researchers cause for the success and level of accuracy of the other three modules. According to my contribution, I trained a module to auto generate profiles for the users who log into the system by considering abstracts from the most cited publications of Google Scholar and using IEEE API by I trained the module by using 800 abstracts from each subject area. To do my task I used Machine learning, Natural Language Processing, Topic Modeling and some text processing techniques and python libraries. The outputs given by me are used in the other modules and mostly in the recommendation part. I contributed myself to give the most reliable and accurate output while being an active member of the group.

Name of the Student: Jayakodi S.M.

The one of the major issues that researcher faces when dealing with research field is finding appropriate journal or conference for publication. Due to the novelty to the field and limited knowledge are some of the reason for that fact. Normally researcher search places to publication through the studying guidelines manually. When publishing manuscript in a journal there are some standards and limitations. Before accepting publication particular journal consider about alignment of the aim and the scope of journal with the manuscript. So researchers have to consider and comply with those standards. Manual method to do this time consuming and difficult. In other 33 hand research papers may be directly accept or reject. Another difficulty is some journals allow publication free of charge but some of them charges for publication. My individual contribution is Recommendation of journals and conferences for manuscript which provide solution to overcome above mentioned difficulties that researchers have to face. Through this module expect to provide recommendation with high quality and accuracy than manual process. Normally when publishing research papers consider relevance of title, scope and aim alignment. As an example,

to improve quality of the recommendation system have to identify about gap between expert researcher and novelty researcher publication.

Name of the Student: Dilrukshi S.A.P.S

Identifying social connectivity of researcher to suggest collaborators for their future work was done through the Social network analysis. Basically, the implementation is done in two main approaches to detect connectivity between researchers. The first approach is Co-authorship networks, in which two researchers are considered connected if they have co-authored one or more scientific papers. This approach visualizes the researcher co-authorship network which is built from the Google scholar data and suggest the strongest connected co-author and the strongest connected co-authors of co-authors. The input of this module is author name and the research paper title and related co-authors names for each and every title. Through the co-authorship network provide suggestions for collaborators. Suggestions are provide based on the level of strong of the relationship that have the author. Here, suggest the strongest connected co-authors and the strongest connected co-authors of co-authors.
Second approach visualizes the researcher Affiliation network and suggests the connectivity of the authors around that related affiliation. The input of this module is the author name and affiliation of each author. Here suggest the collaborators who are in same affiliation.

Name of the Student: Premarathne A.G.M.I

My contribution of this project is to recommend best matching collaborators to a logged user by using the outputs given by the profiling part and social connectivity part. When recommending the top matching collaborators for the newly logged user, the grade of relevance is being considered as well as connectivity because good recommendation based on these two. I follow a hybrid algorithm for getting the output and recommended the best matching collaborators which have at least a small amount of connectivity. I contributed myself for the project thoroughly while working with the team members collaboratively.

Appendix B

LSA Concepts in word range 200

Concepts = 6 , 200 words per concept

{'**Machine Learning Concept**': [('learning', 0.3262424157048821), ('machine', 0.2330346635808693), ('data', 0.21895832306565208), ('method', 0.1634343479710971), ('system', 0.1573001641589981), ('algorithm', 0.15669914050973052), ('model', 0.15500040854267622), ('network', 0.14051468158657268), ('feature', 0.13254681442444918), ('classification', 0.11010533914054003), ('approach', 0.1082441879944546), ('image', 0.1026489146832746), ('technique', 0.09823033805125105), ('problem', 0.09724967038157582), ('proposed', 0.09661977309201975), ('detection', 0.09597813279278422), ('based', 0.09398227276278046), ('set', 0.0927317361623745), ('result', 0.09011992953720649), ('used', 0.0900851182729885), ('training', 0.08931300621341721), ('performance', 0.08843756374723283), ('information', 0.08774708049405151), ('application', 0.08752117220055118), ('neural', 0.08493516457300326), ('vector', 0.08319637083285848), ('svm', 0.07969702561521891), ('support', 0.0796539059620271), ('processing', 0.07952211599661563), ('accuracy', 0.07652596780815919), ('prediction', 0.0764761738419024), ('recognition', 0.07568566376072922), ('deep', 0.07547428417183619), ('classifier', 0.07129589025966906), ('different', 0.06684667150508218), ('research', 0.06648200839385619), ('time', 0.0627045186025731), ('analysis', 0.062559939354901531), ('decision', 0.05980048648388169), ('knowledge', 0.05892984110151231), ('also', 0.0588042679291649), ('process', 0.05722512964680449), ('pattern', 0.05639385717597197), ('show', 0.056381751196508914), ('function', 0.05293334387724457), ('one', 0.052873086552932394), ('two', 0.05250482907746459), ('framework', 0.05242944724206902), ('many', 0.05218524320617071), ('structure', 0.05112601052531591), ('human', 0.05043019966379604), ('computer', 0.04982378573904119), ('extreme', 0.04980305834865794), ('number', 0.04975315076703123), ('theory', 0.04890645848839953), ('sample', 0.04836204242025137), ('design', 0.04821673180190248), ('control', 0.04774683985513017), ('however', 0.04739228172005806), ('large', 0.04715986445248528), ('technology', 0.04653305174016890), ('applied', 0.04495361249302088), ('various', 0.04391880259384822), ('input', 0.043432513025950366), ('kernel', 0.04282688112127628), ('big', 0.04262995949326941), ('signal', 0.04233847630136937), ('object', 0.042294051292862324), ('high', 0.04212574426407563), ('architecture', 0.04183438090307228), ('several', 0.04173605722263144), ('domain', 0.04165321902858254), ('intrusion', 0.04157598980005226), ('area', 0.04142220151228869), ('field', 0.04124097244583554), ('example', 0.040934220560412404), ('parameter', 0.04076735976468643), ('computational', 0.040701593347766324), ('intelligence', 0.040317093505757316), ('language', 0.04027394042038444), ('solution', 0.039836807798496264), ('artificial', 0.039765748261002286), ('tree', 0.039593886405149124), ('test', 0.03953825439982557), ('experiment', 0.03946611444060411), ('space', 0.03920301876124311), ('extraction', 0.039045977588692916), ('modeling', 0.03875683123068151), ('ensemble', 0.038729691661231816), ('learn', 0.03868512571365479), ('experimental', 0.038654944297844336), ('well', 0.03859631798046214), ('propose', 0.03858948603677933), ('fault', 0.038377512229340896), ('optimization', 0.038286688814816686), ('case', 0.038283462628570034), ('database', 0.03820328419130753), ('first', 0.03794481132735539), ('power', 0.03751938726314448), ('selection', 0.03717070503979424), ('important', 0.0369970364816371), ('environment', 0.036795317699247 32), ('developed', 0.036777510186526155), ('compared', 0.03676720957348279), ('fuzzy', 0.036765406006468146), ('regression', 0.03645005294629521), ('traditional', 0.03644449442139374), ('computing', 0.03631273445008413), ('datasets', 0.03627958808336969), ('diagnosis', 0.03623701658320955), ('mining', 0.036180971745177565), ('topic', 0.03614871366130971), ('error', 0.0359751474034531), ('face', 0.03596437725552992), ('intelligent', 0.03572752356746309), ('novel', 0.03562941017736654), ('security', 0.03561325700992057), ('better', 0.03533322379830269), ('tool', 0.0352635635610094), ('provide', 0.0350641909316501), ('type', 0.03489096792708856), ('existing', 0.03488867609503615), ('statistical', 0.034794460557118545), ('make', 0.034774756403315364), ('sensor', 0.034396588159617464), ('point', 0.0340394031071536), ('supervised', 0.03354849702737376), ('bayesian', 0.03329993415841111), ('quality', 0.03298326708651446), ('order', 0.032922673728417604), ('ability', 0.03279771009028336), ('class', 0.03262996582168159), ('task', 0.03259644731694295), ('improve', 0.03252530021742254), ('representation', 0.03210255718391414), ('may', 0.032003138322460004), ('challenge', 0.03193105689952192), ('behavior', 0.031805900635686335), ('online', 0.031425499007907566), ('issue', 0.031403031047119494), ('characteristic', 0.031383793620973814), ('output', 0.03136729249207953), ('vision', 0.03130764227960792), ('effective', 0.031300441238666855), ('way', 0.03097070214294972), ('elm', 0.03088329031647916), ('attack', 0.03076755011409415), ('text', 0.030606240915263937), ('including', 0.030340228701620595), ('search', 0.030323511349844953), ('need', 0.03029386315373458), ('rule', 0.030263824396701017), ('recent', 0.030072359342257656), ('concept', 0.030043122077393066), ('scheme', 0.030005671968335774), ('obtained', 0.029941432261002477), ('implementation', 0.029854816240609795), ('presented', 0.029686972006349285), ('dynamic', 0.029613976240663774), ('due', 0.029494533653938317), ('memory', 0.029453473352535234), ('speech', 0.02926188123728971), ('given', 0.029218311678593203), ('level', 0.02905252576502334), ('term', 0.029041660153271656), ('development', 0.0289908726250681), ('robot', 0.02868212341405797), ('complex', 0.028679372593586067), ('estimation', 0.02855095133443305), ('unsupervised', 0.02825901840374067), ('layer', 0.02788264097608664), ('value', 0.027840822485159407), ('multiple', 0.027804951998377413), ('provides', 0.02759982688371094), ('software', 0.027494091564298023), ('three', 0.027439324137669953), ('rate', 0.027368873673423607), ('capability', 0.027044014023266675), ('key', 0.027003585826399057), ('strategy', 0.02699556028920987), ('give', 0.026907746248373148), ('evaluation', 0.026872739831923374), ('without', 0.026754237275610437), ('dataset', 0.026735259435745833), ('like', 0.02666950100074278), ('present', 0.02662654260140349), ('cloud', 0.02652626755186918), ('effectiveness', 0.02649430862970476), ('real', 0.02636358987608105), ('shown', 0.02624552092938442), ('amount', 0.02620188039417096), ('finally', 0.02601110665275367), ('semantic', 0.026008163306737945), ('interaction', 0.02591731623435816), ('retrieval', 0.025906885131356167), ('management', 0.025881184114175924), ('source', 0.02586520557784948), ('device', 0.02573861015692303), ('engineering', 0.02556048732013106), ('learner', 0.025417818229366656)]}

'**Big Data Concept**': [('data', 0.5701386492053666), ('big', 0.3484741095235427), ('system', 0.1430923007570735), ('cloud', 0.10847740550427735), ('network', 0.10769606674000516), ('analytics', 0.10623208977305436), ('application', 0.10181245159835453), ('technology', 0.0976007399056455), ('computing', 0.09622986974658022), ('processing',

0.09130476848834904), ('analysis', 0.08974890990820109), ('model', 0.08452801594660508), ('information', 0.08372749671727268), ('challenge', 0.0799313575109589), ('research', 0.07305954896222314), ('proposed', 0.06816079573077688), ('algorithm', 0.06686235386077208), ('method', 0.06599996540221294), ('based', 0.06570302040575912), ('also', 0.06375190011087925), ('architecture', 0.06313686213993248), ('storage', 0.06271747407743553), ('security', 0.06188299821181848), ('privacy', 0.061677668022724015), ('business', 0.061308057921747615), ('framework', 0.060809768737020216), ('management', 0.06079318986078498), ('problem', 0.05977335657209399), ('different', 0.05937177297279605), ('performance', 0.0591235314780139), ('large', 0.058286946711638116), ('process', 0.05826715400683511), ('technique', 0.05788232227315351), ('used', 0.05697711972604604), ('approach', 0.056358797381611384), ('learning', 0.056013740527335446), ('power', 0.0555497584595623), ('service', 0.055450518970383715), ('result', 0.05508808417387432), ('volume', 0.054622048235089 13), ('platform', 0.05190082875832748), ('time', 0.05148064473336111), ('social', 0.0513993028412379), ('development', 0.05075748846028614), ('smart', 0.050483938289409225), ('amount', 0.05007820395230518), ('set', 0.05006228955602333), ('many', 0.050039843912647916), ('issue', 0.0493410069203029), ('healthcare', 0.04906691477576028), ('high', 0.04823666360495903), ('distributed', 0.047611450955969005), ('quality', 0.04742958302155044), ('solution', 0.04713399766206734), ('various', 0.046718497112344416), ('tool', 0.04671780794884134), ('mining', 0.0456172054841432), ('provide', 0.045117588670405724), ('source', 0.04470360737428552), ('however', 0.04281735629428369), ('value', 0.04250476119225925), ('traditional', 0.04244293801177105), ('device', 0.04236612280273716), ('internet', 0.04197181374534015), ('knowledge', 0.04189204328769892), ('sensor', 0.04113269227078136), ('important', 0.04103540892478537), ('one', 0.04079668176371764), ('resource', 0.04002306720339686), ('need', 0.039957026709083306), ('design', 0.03980623761246639), ('hadoop', 0.03916723739840788), ('industry', 0.03916247594556656), ('database', 0.038967217015283745), ('iot', 0.03799167934697355), ('environment', 0.037866162930982064), ('feature', 0.03777840068511006), ('traffic', 0.03763059903597016), ('science', 0.037379979060899036), ('characteristic', 0.037199948647074876), ('health', 0.03671417153595765), ('show', 0.03641196042878751), ('variety', 0.036258320545993886), ('area', 0.036161722222308346), ('requirement', 0.036108932120952564), ('several', 0.03495487472502777), ('support', 0.034914486272388204), ('machine', 0.03490312430736214), ('decision', 0.034480393089138896), ('term', 0.034441917756606925), ('propose', 0.03432844065148061), ('make', 0.03426457965164044), ('mobile', 0.03418865998469272), ('order', 0.03375781249682822), ('well', 0.03365429202473594), ('first', 0.03357101172365395), ('grid', 0.03344355608951505), ('software', 0.032921722950406546), ('type', 0.03283290432196645), ('two', 0.03267726864466976), ('field', 0.03265526229424785), ('number', 0.0325043624342064), ('organization', 0.03244042043905583), ('user', 0.03230822712016054), ('domain', 0.03219516382888175), ('future', 0.03210952801314743), ('efficient', 0.03206293743364513), ('discus', 0.03194795809616117), ('scheme', 0.031788908008754536), ('huge', 0.03164183621937577), ('current', 0.0313155994251183), ('medical', 0.03123938035574213), ('case', 0.03095261523707244), ('potential', 0.030952365335864013), ('due', 0.030862568963727792), ('key', 0.030353319164428936), ('pattern', 0.03014723707190023), ('realtime', 0.029987988729612673), ('recent', 0.029885610524442767), ('including', 0.02988504826694765 2), ('structure', 0.02950508913983605), ('level', 0.029485001873211052), ('generated', 0.029425028507037366), ('operation', 0.029353608065296338), ('benefit', 0.029270418435526104), ('world', 0.029059593065492 27), ('collection', 0.028887470139236218), ('size', 0.02867122200962863), ('related', 0.028524369004914216), ('way', 0.028475947047506354), ('unstructured', 0.02845860915097075), ('heterogeneous', 0.028350658858504036), ('increasing', 0.028282154688311373), ('infrastructure', 0.028248425133141578), ('clustering', 0.02799975863534495), ('developed', 0.027989554100110756), ('focus', 0.027832601185730187), ('velocity', 0.02775242615200616), ('become', 0.027684068534863656), ('communication', 0.02767160644028491), ('analyzing', 0.027620871112545832), ('like', 0.027454008524333013), ('year', 0.027350691400747 11), ('massive', 0.027342780327537484), ('address', 0.027306682692935482), ('era', 0.02729726122534432), ('may', 0.027296489652776545), ('presented', 0.02720322563696556), ('finally', 0.027069136111599255), ('opportunity', 0.027005154091336844), ('access', 0.026992295778323048), ('novel', 0.02690042152731), ('energy', 0.02688568598160024), ('monitoring', 0.026740953228461782), ('thing', 0.02660057384211554), ('complex', 0.026317058882845704), ('mechanism', 0.026243595785011516), ('cluster', 0.02620852573332977), ('center', 0.02618629174464759), ('digital', 0.026153930670173735), ('provides', 0.026067287853782848), ('dimension', 0.02594386908453543), ('query', 0.025925996389641548), ('medium', 0.02585471994238452), ('control', 0.02583035726378916), ('better', 0.025782735987599 54), ('analyze', 0.02568502562171779), ('classification', 0.025677670857289343), ('intelligent', 0.025616086785897746), ('multiple', 0.025584933552905992), ('efficiency', 0.025572276622334963), ('improve', 0.025538875893252072), ('existing', 0.025238971869392057), ('increase', 0.025043185570401343), ('integration', 0.02496402624381341), ('form', 0.024926986304107696), ('effective', 0.024683176072409673), ('wireless', 0.02465755403200245), ('cost', 0.024539799946903463), ('help', 0.024539443421503684), ('role', 0.024469589362375005), ('capability', 0.024449173337244296), ('structured', 0.02419321777383541), ('mapreduce', 0.023904511831691882), ('implementation', 0.023872031649420138), ('thus', 0.02385783063117303), ('component', 0.02370269301026016), ('among', 0.02369136857472005), ('manage', 0.023645430688491363), ('company', 0.023640100603613115), ('could', 0.023618808944296372), ('reduce', 0.023549467811110326), ('handle', 0.02345641106251697), ('real', 0.023414565209197477), ('making', 0.02337233360141381), ('concept', 0.023345137641499397), ('sensing', 0.023195686397405345), ('experiment', 0.023169433433974373), ('collected', 0.02307265460102714), ('available', 0.02307154680203633)]}

'**Data Mining Concept**': [('data', 0.4774907117903035), ('mining', 0.3185506949742487), ('algorithm', 0.15883378931260728), ('system', 0.1544320962919936), ('web', 0.12372581129417673), ('rule', 0.11987479759763472), ('information', 0.11813929915034811), ('method', 0.11698191659410763), ('pattern', 0.11307702759513433), ('knowledge', 0.11145607471114603), ('technique', 0.10623774671335595), ('process', 0.10031918602293256), ('network', 0.09688400773948498), ('analysis', 0.09616616474782713), ('application', 0.09363746763455401), ('model', 0.09325124740112493), ('database', 0.09267510421595972), ('set', 0.08730510618936588), ('used', 0.08094423843959815), ('result', 0.08030082569398544), ('research', 0.0796018851465839), ('approach', 0.07869980173438207), ('based', 0.07756547256951878), ('learning', 0.07358562857459564), ('proposed', 0.0716287488086329), ('clustering', 0.07137817069409072), ('problem', 0.0710077285445722), ('association', 0.07054724461419623), ('computing', 0.07037655416317615), ('technology', 0.06715808122139158), ('classification', 0.06659407455383735), ('feature', 0.06448169231953955), ('management', 0.06414395976539461), ('frequent', 0.06366151454567424), ('processing', 0.06332476027383882), ('also', 0.06288377820983133), ('topic', 0.06258729925882262), ('fuzzy', 0.062431062311000755), ('big', 0.06116339129304346), ('large', 0.0611250003850684), ('time', 0.06068072538259539), ('spatial', 0.0586727508520398),

('different', 0.05712484778552074), ('discovery', 0.05440934527080116), ('privacy', 0.05264442218393678), ('important', 0.05237386238494851), ('security', 0.05142142893805972), ('performance', 0.051381842983942244), ('detection', 0.05066566110953121), ('many', 0.05003991102990305), ('support', 0.04994141235587801), ('distributed', 0.04925948524309674), ('structure', 0.04864305919067989), ('one', 0.04850178248912129), ('decision', 0.04812456566525219), ('tool', 0.047451092358670524), ('following', 0.04730007774800233), ('image', 0.04695239536080964), ('framework', 0.04676082868856884), ('mine', 0.04653008598184896), ('field', 0.04618084557807492), ('dealt', 0.04544928478209994), ('user', 0.04403546065283374), ('stream', 0.0437693797763655), ('software', 0.043571434224677366), ('useful', 0.04288872777899824), ('type', 0.042078516996097985), ('show', 0.041906483418425254), ('social', 0.04151474645964229), ('however', 0.041295400091639026), ('environment', 0.04082841265432134), ('development', 0.04060275441139438), ('need', 0.04053563888320051), ('various', 0.0403637768591369), ('text', 0.040346005888554436), ('query', 0.039995722504128736), ('machine', 0.03996274870564998), ('engineering', 0.03986837231215175), ('source', 0.039641873711336276), ('high', 0.039196236120932995), ('domain', 0.03915905504777825), ('area', 0.03913164188337832), ('intelligent', 0.039037036943541104), ('medical', 0.03868109214821974), ('provide', 0.038668667452471134), ('order', 0.038248296782777455), ('sensor', 0.037972654382422706), ('two', 0.03787432824267233), ('business', 0.03780134705105916), ('neural', 0.03740659068791068), ('visualization', 0.03714547879581216), ('existing', 0.036853538614093614), ('amount', 0.03669943027504855), ('number', 0.036268800292892776), ('internet', 0.036093647880177666), ('efficient', 0.0360136366513656), ('well', 0.03543648617006893), ('issue', 0.035320908122919395), ('cluster', 0.03526934155309171), ('item', 0.034955348861894525), ('first', 0.03493614270958147), ('sensitive', 0.034756114657458176), ('make', 0.034751531787186256), ('may', 0.034748410437218116), ('quality', 0.03465571629152883), ('intelligence', 0.034210206911874895), ('warehouse', 0.033701019612908105), ('real', 0.033339786584533584), ('value', 0.03321858766843448), ('effective', 0.03313278909381848), ('propose', 0.03303012040329617), ('computer', 0.03293958269181652), ('challenge', 0.03286139584686055), ('applied', 0.03265332557580582), ('several', 0.03197711611581083), ('educational', 0.03188850984544641), ('tree', 0.0316165884844762), ('intrusion', 0.03145822892310581), ('extraction', 0.031223861090124497), ('architecture', 0.03096604342038478), ('transaction', 0.030928994717429038), ('novel', 0.030649825264886566), ('event', 0.030212189843506095), ('accuracy', 0.030184627110199768), ('improve', 0.03002032135445484), ('prediction', 0.02998479709855884), ('experiment', 0.02991891401717841), ('hidden', 0.029572729115156432), ('developed', 0.029239182312768886), ('object', 0.029197947730667338), ('extract', 0.029066201259289565), ('representation', 0.028916128696709675), ('given', 0.02886670517790069), ('log', 0.02859330010637676), ('function', 0.028530004208513076), ('multimedia', 0.028417260408051655), ('grid', 0.028416378860973337), ('including', 0.028403035315663808), ('complex', 0.028338753885507016), ('mobile', 0.028155738441530444), ('graph', 0.028065627746544704), ('traditional', 0.02800499523744224), ('recognition', 0.027976500526515423), ('experimental', 0.027852770684947535), ('better', 0.027851365739711535), ('power', 0.0278432926089783), ('modeling', 0.027726570476037937), ('datasets', 0.027691203063477617), ('three', 0.02767419927860735), ('selection', 0.027426073846163735), ('concept', 0.02716730862447767), ('researcher', 0.027141060181949037), ('operation', 0.027124318153539516), ('find', 0.027105728690901824), ('current', 0.026858971491723026), ('artificial', 0.026854804799370063), ('focus', 0.02674706550399348), ('interaction', 0.026736744862384264), ('attribute', 0.026687477570700436), ('solution', 0.026609792716399044), ('science', 0.02652585242693812), ('efficiency', 0.026512334151392966), ('parameter', 0.026462651392266976), ('called', 0.02645482342756138), ('sequential', 0.02643063258823616), ('presented', 0.02614132808328687), ('space', 0.026134380787241257), ('discus', 0.026049072618633623), ('term', 0.025898753379551654), ('usage', 0.02577380792775215), ('recent', 0.0256497155474503), ('point', 0.02561988523549445), ('example', 0.02530580999045833), ('way', 0.025264806960081484), ('search', 0.02524724650983334), ('discover', 0.025148691975693995), ('itemsets', 0.02507982059665142), ('series', 0.025063401580351908), ('case', 0.025059688857315757), ('retrieval', 0.024956861005921302), ('monitoring', 0.024862094858065304), ('agent', 0.024778947200698707), ('parallel', 0.024376443562404627), ('relationship', 0.02431676056823891), ('language', 0.02427807192129259), ('behavior', 0.024276250925641642), ('related', 0.024232604928558325), ('preprocessing', 0.024049838755606904), ('computational', 0.02398617861986029), ('present', 0.023936254015967967), ('mined', 0.023904947012356913), ('theory', 0.02390399363199792), ('dynamic', 0.02377441702841329), ('human', 0.023769181800187764), ('component', 0.023718688014130476), ('huge', 0.0235872407570325), ('kind', 0.023475797383461363), ('specific', 0.02343170009735036), ('help', 0.02343532093006284), ('resource', 0.023413019463318887)]}

'**Computer Vision Concept**': [('image', 0.23645182733186718), ('vision', 0.2100097257464748), ('system', 0.1999133449498403), ('object', 0.19183708072917324), ('computer', 0.1857789255953884), ('method', 0.1557445059731008), ('model', 0.1522863012535426), ('algorithm', 0.14251761065745094), ('application', 0.1232107425894621), ('feature', 0.12068514933979718), ('recognition', 0.11879654254359222), ('camera', 0.1135779522992326), ('processing', 0.11101950367743944), ('problem', 0.10790806856565298), ('approach', 0.10002766675657028), ('technique', 0.09961523628691858), ('human', 0.09715861073824744), ('based', 0.09636895713546528), ('face', 0.09567250413891863), ('data', 0.09555691614386348), ('reconstruction', 0.09537113772306723), ('information', 0.0917232783237387), ('scene', 0.0913642072594439), ('tracking', 0.086856723368281), ('point', 0.084766099187567), ('video', 0.08367166226035937), ('proposed', 0.0828190902328683), ('detection', 0.082010098717044), ('used', 0.08164262660580623), ('pose', 0.08126530792284888), ('motion', 0.08083861399714772), ('result', 0.08065211975281253), ('shape', 0.0727800537623546), ('estimation', 0.07241729253172775), ('network', 0.06957344571864799), ('surface', 0.067837626217081), ('performance', 0.06688635343120386), ('visual', 0.06688361935041473), ('many', 0.06686426331920789), ('analysis', 0.06665889560943093), ('field', 0.06597505010660204), ('depth', 0.06558902447759266), ('learning', 0.06518097708784511), ('sensor', 0.06506919863617919), ('one', 0.06445733991968841), ('research', 0.062014287072974884), ('robot', 0.061865103096182664), ('different', 0.06143387546840172), ('also', 0.0609449311259072), ('two', 0.06055011103657645), ('stereo', 0.05909167811032391), ('process', 0.05688477203056017), ('matching', 0.05648749926194887), ('show', 0.05572121855739019), ('time', 0.054666209945898776), ('control', 0.05452414795720694), ('technology', 0.05393298103817026), ('view', 0.053234202768956504), ('important', 0.05188255818463623), ('set', 0.051422845574274935), ('environment', 0.05043291422661777), ('range', 0.04991268432038099), ('graphic', 0.04980974389598897), ('real', 0.049540440662795046), ('segmentation', 0.049016450861551784), ('representation', 0.048682808720213605), ('framework', 0.04778730463783575), ('interaction', 0.046550633663609516), ('solution', 0.046004380521536097), ('realtime', 0.04563328370734474), ('parameter', 0.04557309122050918), ('development', 0.045457873563553686), ('accuracy', 0.045322158189894865), ('architecture', 0.04522014430915851), ('several', 0.045184478510853054), ('machine',

0.04439491789156437), ('number', 0.044367241724507386), ('topic', 0.04422978590901277), ('propose', 0.04352130781733353), ('multiple', 0.04308005381227994), ('developed', 0.04202870678853874), ('sequence', 0.041986883920812346), ('high', 0.04177872238687433), ('gt', 0.041745697829270505), ('various', 0.04163080935054571), ('classification', 0.04139412375723803), ('well', 0.04134982751836888), ('modeling', 0.041123588509793356), ('design', 0.04058963620682864), ('large', 0.04054864788898072), ('hand', 0.040313565948232474), ('device', 0.04002686208198342), ('pattern', 0.039866464203580046), ('however', 0.039858207857278304), ('space', 0.039771847050003355), ('area', 0.039685435351028364), ('experiment', 0.03938131697688208), ('task', 0.039291230786328415), ('robust', 0.03911529775202175), ('neural', 0.038718466389678054), ('facial', 0.03852096931313681), ('first', 0.03851124101998544), ('function', 0.03847334049321303), ('structure', 0.03816192682008227), ('order', 0.03808171435330911), ('target', 0.037891708659311366), ('position', 0.03761558096532797), ('presented', 0.037419136508648525), ('vehicle', 0.03739034634897745), ('color', 0.036694600193090315), ('deep', 0.036632281748652185), ('hardware', 0.03630855784009502), ('head', 0.03602189360172582), ('single', 0.035747148411184565), ('signal', 0.03568543783631541), ('registration', 0.03556890868132708), ('accurate', 0.03516435982403364), ('map', 0.03507476683771744), ('digital', 0.03496436124613236), ('experimental', 0.03467087723716228), ('distance', 0.03453414587453964), ('frame', 0.034377598757392554), ('understanding', 0.034203810827127035), ('computational', 0.034044804482316955), ('following', 0.03357859367935426), ('novel', 0.03349236447353962), ('correspondence', 0.033396719732763674), ('recent', 0.03318232624907168), ('three', 0.033088524928556066), ('edge', 0.03304919331765853), ('gesture', 0.032798567063656615), ('knowledge', 0.03276595667357821), ('obtained', 0.03257247055750699), ('estimate', 0.03249642519514163), ('local', 0.03218619155956904), ('imaging', 0.032035366287562414), ('error', 0.03201423379013908), ('virtual', 0.0317644056924722), ('region', 0.03171637747863731), ('body', 0.03168566576412529), ('example', 0.03155812722652205), ('part', 0.03148363821838119), ('possible', 0.03147561715583006), ('quality', 0.031460552625120486), ('make', 0.031460070641679244), ('moving', 0.03127660970083347), ('existing', 0.031247271904458272), ('level', 0.031231746485043156), ('interest', 0.03116681692927438), ('geometric', 0.031125222067397224), ('interface', 0.03086535129645435), ('implementation', 0.030741192240075908), ('robotics', 0.030734405667092363), ('key', 0.03053970571597946), ('current', 0.030535644432442968), ('parallel', 0.030336822929559727), ('training', 0.030230767553665206), ('measurement', 0.03009955416850549), ('geometry', 0.03003249488935322), ('describe', 0.029993780117891784), ('input', 0.029954280001624815), ('achieve', 0.029904428870224018), ('artificial', 0.02989873851543342), ('us', 0.029643443302297613), ('software', 0.029587003498331484), ('computing', 0.02952517607690557), ('complex', 0.02951857229017246), ('type', 0.029513857603630844), ('challenging', 0.029500210470294223), ('present', 0.029276857002917333), ('provide', 0.029236606538792842), ('demonstrate', 0.029195913702298452), ('need', 0.028882253272695085), ('operation', 0.02886022113543974), ('extraction', 0.028827362048711617), ('constraint', 0.028776985218781477), ('vector', 0.02867284563510179), ('including', 0.02862781810699662), ('given', 0.028584598301897093), ('descriptor', 0.028506735676389457), ('reality', 0.028490756737830965), ('goal', 0.02847082043153604), ('action', 0.02840297526781747), ('dealt', 0.028393860838389885), ('database', 0.028214985560779032), ('applied', 0.02816913789802651), ('extract', 0.02816532193203645), ('sparse', 0.02804894156485224), ('cost', 0.027951937147117138), ('form', 0.027827499796701165), ('provides', 0.027622909062743943), ('challenge', 0.027620667918618006), ('step', 0.027542343978796117), ('advantage', 0.02749746622213049), ('due', 0.027280163658657965), ('concept', 0.02724727420495372 6), ('automatic', 0.02717255971165511 2), ('computation', 0.027001029809053424), ('available', 0.02693670065933573), ('issue', 0.026687278103482467)]}

'**Bioinformatics Concept**': [('data', 0.2550597330113166), ('bioinformatics', 0.19391720084914366), ('sequence', 0.18265166049315748), ('system', 0.1642089759696012), ('algorithm', 0.16187649116866443), ('protein', 0.13559024459278401), ('method', 0.13208542172850296), ('application', 0.13140613608258936), ('computing', 0.12935289416717433), ('problem', 0.12779301273683408), ('grid', 0.11896032265665786), ('gene', 0.11358961803082797), ('analysis', 0.11319596276726776), ('biological', 0.1086967546513046), ('information', 0.10757369498225323), ('network', 0.10527828393612998), ('tool', 0.10111892839335655), ('processing', 0.09964174026813293), ('model', 0.09880788796892893), ('computational', 0.09425097211142426), ('approach', 0.09183496247711827), ('database', 0.09017069521400589), ('result', 0.08923462400384057), ('learning', 0.08895195438699101), ('alignment', 0.08891983803355727), ('research', 0.08792306086157312), ('based', 0.08212015765704282), ('used', 0.08199411042972583), ('performance', 0.08140286357971252), ('technology', 0.0807564681088458), ('resource', 0.07931260692137067), ('technique', 0.07838195930330535), ('workflow', 0.07812332149769924), ('mining', 0.07727628419653179), ('proposed', 0.07395465869729834), ('parallel', 0.07155337627376801), ('machine', 0.07077116460568432), ('structure', 0.07050186891219605), ('feature', 0.0701645966764669), ('one', 0.06893418949856792), ('dna', 0.06881603937742371), ('software', 0.06790563970103533), ('biology', 0.06753848616624052), ('clustering', 0.06735589244858346), ('time', 0.06587379261266657), ('set', 0.06539020656417516), ('important', 0.06490157212351133), ('large', 0.064293460548451), ('web', 0.062312874192381545), ('science', 0.061356550546644836), ('field', 0.06068614769137182), ('pattern', 0.05942483023559882), ('many', 0.05850097854131824), ('also', 0.05821754277051337), ('classification', 0.058020561286282094), ('distributed', 0.05753751840916038), ('process', 0.057444320718418714), ('multiple', 0.057174744417850955), ('computer', 0.057102442984905605), ('complex', 0.05662688027915258), ('prediction', 0.05581457927550668), ('environment', 0.05535489368651341), ('different', 0.05529500688073782), ('architecture', 0.053585441504212294), ('image', 0.053061818471419275), ('framework', 0.052959085629197315), ('number', 0.05238413461698633), ('topic', 0.052360577979490994), ('solution', 0.05208625887229164), ('genome', 0.05148382513187061), ('show', 0.051092357149574656), ('service', 0.050489832754051944), ('high', 0.05012059912607485), ('knowledge', 0.04955188709702257), ('cluster', 0.049515771981576556), ('search', 0.04943831520968211), ('implementation', 0.04919665903368081), ('genomic', 0.04890481567014460 4), ('design', 0.04846610284143903), ('development', 0.04821912479959857), ('genetic', 0.047354916145543526), ('area', 0.04732344232050792), ('two', 0.047165446788417595), ('experimental', 0.04709284249532555), ('management', 0.04691535647420124), ('following', 0.046546183377628 58), ('experiment', 0.04550729000516662), ('datasets', 0.04535781163531305), ('several', 0.04519132298094372), ('engineering', 0.045042575014997165), ('sequencing', 0.04351550131773146), ('support', 0.04309333614030788), ('existing', 0.042864183109871015), ('function', 0.04277281908943831), ('various', 0.042713014523893755), ('dealt', 0.04270973992197449), ('graph', 0.042644325598967144), ('order', 0.04263491814417825), ('signal', 0.04245884562247551), ('available', 0.04199873283173898), ('ontology', 0.041917621917973595), ('cloud', 0.04187636090057919), ('security', 0.04186522697037 06), ('however', 0.04170623701080283), ('challenge', 0.04152871432387656), ('efficient', 0.04148872877915589), ('novel', 0.041451774886201935), ('domain', 0.04129503009420263), ('scientific',

0.041149168333455804), ('developed', 0.0410597065666727), ('molecular', 0.04095165222688309), ('need', 0.04058490734372218), ('source', 0.03991150995247835), ('tree', 0.039737741182599255), ('expression', 0.039656350495402344), ('platform', 0.03965035021075324), ('well', 0.0395752371720069), ('provide', 0.038963744778545896), ('applied', 0.038851321080154155), ('amount', 0.03834795940074227), ('execution', 0.03817700777336666), ('recognition', 0.03807646388378893), ('modeling', 0.03760095846618456), ('project', 0.03737508176461827), ('selection', 0.03725571387806432), ('heterogeneous', 0.03724789449440145), ('computation', 0.03657773692873367), ('intelligent', 0.03641588562831055), ('query', 0.03626238071212584), ('access', 0.03625323459338638), ('provides', 0.03584919022383533), ('intelligence', 0.0356118671569805), ('power', 0.03558409041718063), ('integration', 0.035317294767687205), ('similarity', 0.03502159174861531), ('issue', 0.03486198492650371), ('interaction', 0.03482360226113829), ('big', 0.034392911302048065), ('accuracy', 0.03430178692392091), ('medical', 0.0341639593234216), ('interface', 0.0340207111253907), ('program', 0.03388285846111591), ('classifier', 0.03361188916808318), ('propose', 0.03350206581347614), ('presented', 0.03318455931396949), ('disease', 0.03315772796426557), ('real', 0.03299653969946317), ('component', 0.032707225983339215), ('researcher', 0.0325452646604978), ('including', 0.03245091942013857), ('motif', 0.03201326101760335), ('compared', 0.031951733371241756), ('dynamic', 0.03190159542381202), ('life', 0.03160280665806281), ('biomedical', 0.03129523426573074), ('analyze', 0.031285202805064756), ('programming', 0.0312791134325672), ('human', 0.031219767074446708), ('communication', 0.030876036094750564), ('discovery', 0.030430423365583572), ('neural', 0.03037848343911995), ('simulation', 0.03036331366448922), ('language', 0.030282603662902738), ('way', 0.030264621384183704), ('first', 0.03004020042428753), ('finding', 0.029888651330257598), ('given', 0.029810233528904016), ('understanding', 0.02974447835660374), ('community', 0.0296433597623697), ('role', 0.029587073772667706), ('related', 0.02957746762692663), ('example', 0.02940913610377224), ('identify', 0.02932420211295108), ('biologist', 0.029086732928200135), ('msa', 0.029075598403061443), ('three', 0.02887788822501614), ('artificial', 0.028802621103616966), ('memory', 0.028773819317575573), ('better', 0.02872982807350125), ('present', 0.028592894824561805), ('specific', 0.02847100982842705), ('visualization', 0.028275084406226957), ('control', 0.028197667143670935), ('scheduling', 0.0280276572785019), ('recent', 0.027820278558592763), ('microarray', 0.02781910309579754), ('solve', 0.027808458718294855), ('current', 0.027676317810605068), ('make', 0.027486751844870778), ('test', 0.02747745761555895), ('called', 0.02741498165441457), ('among', 0.027365889171406824), ('storage', 0.02726305427763012), ('requirement', 0.027109557852803244), ('find', 0.02700761946295495), ('speedup', 0.026957958723869514), ('evolutionary', 0.02692609561183354), ('scientist', 0.026843786506073965), ('wireless', 0.026658481484014895), ('scale', 0.026626561041240168)]}

'**Artificial Intelligence Concept**': [('system', 0.3021707814994532), ('intelligence', 0.20740024069180577), ('artificial', 0.18476235214669867), ('network', 0.14377975655876526), ('ai', 0.14243084397407593), ('data', 0.13948410835245192), ('model', 0.11371726619006718), ('gt', 0.11333610474286654), ('problem', 0.11303778933971916), ('knowledge', 0.11172389103936256), ('application', 0.11146886734209249), ('method', 0.11117443292003427), ('learning', 0.10814734614994906), ('technology', 0.10486093485608548), ('technique', 0.10364111183028345), ('control', 0.10348950076570439), ('information', 0.10305477105842821), ('algorithm', 0.10038755622828734), ('test', 0.09852315338264532), ('based', 0.09849434597451745), ('intelligent', 0.09744500587724948), ('research', 0.0968223513090244), ('human', 0.09443972980645918), ('computer', 0.09335492597771204), ('design', 0.09310197709636514), ('approach', 0.09158851787245736), ('expert', 0.08764539685562421), ('development', 0.0831616145852074), ('power', 0.08184173281728592), ('neural', 0.08051580121977378), ('robot', 0.08013347058679618), ('process', 0.07803027920111888), ('software', 0.07536916456041762), ('used', 0.07516979618729407), ('proposed', 0.07182240880039967), ('processing', 0.07166501228723875), ('machine', 0.07037564926918102), ('management', 0.06909557184861417), ('architecture', 0.06904840546218063), ('game', 0.06783841614154769), ('cognitive', 0.06769223846360382), ('image', 0.06716894907756032), ('engineering', 0.06592035599812086), ('environment', 0.0651850966587421), ('field', 0.06488230306851164), ('analysis', 0.06448728919772251), ('time', 0.06215061233154618), ('computing', 0.06193909028763796), ('science', 0.06184813884844414), ('feature', 0.06046600543923606), ('also', 0.05898588116761114), ('different', 0.05889569757408891), ('result', 0.0579751679634792), ('theory', 0.057925002044282616), ('decision', 0.05735357345669343), ('ltetx', 0.0573527131373526), ('reasoning', 0.057291097390692845), ('lt', 0.05638642887836409), ('concept', 0.054330626130685795), ('one', 0.052865048819058895), ('distributed', 0.052692303107565217), ('agent', 0.05205492016529864), ('many', 0.051959457870721336), ('program', 0.0519032862147228), ('topic', 0.05178402360379782), ('fault', 0.05144085762879769), ('area', 0.05104026237417326), ('simulation', 0.05079265979453294), ('tool', 0.0495259647248932), ('developed', 0.04890543987283513), ('performance', 0.048298995560278835), ('support', 0.047066457823373695), ('planning', 0.046821840998020496), ('language', 0.04652529614863904), ('following', 0.04518508558562468), ('mobile', 0.04510823351242578), ('diagnostic', 0.044633001037908106), ('communication', 0.04403601855843971), ('solution', 0.044029859710662506), ('computational', 0.044013604838769385), ('future', 0.04373927340357274), ('presented', 0.04361046207135394), ('author', 0.04343108625017191), ('set', 0.04309269627925753), ('world', 0.04224346914743443), ('complex', 0.04207307923514996), ('diagnosis', 0.04202277129980759), ('web', 0.04178800118064731), ('two', 0.04178106204967521), ('level', 0.04170758218682608), ('discussed', 0.04160304592748922), ('dealt', 0.041430542016370864), ('behavior', 0.041410837351569196), ('applied', 0.04072393097902201), ('recognition', 0.04062002890178527), ('capability', 0.04054235020944376), ('signal', 0.04012158323289534), ('search', 0.03996793844904213), ('important', 0.03978269066166249), ('provide', 0.039640890862465666), ('implementation', 0.039300090880754736), ('well', 0.038705297497357835), ('however', 0.03852841673615673), ('project', 0.0384823882818488), ('fuzzy', 0.038348001491260206), ('described', 0.03802359760803627), ('smart', 0.0379950217465826), ('manufacturing', 0.037748287032256016), ('circuit', 0.037062938514720443), ('understanding', 0.037031896849501374), ('discus', 0.036874607266589664), ('first', 0.03682326240278584), ('various', 0.036757698863631665), ('strategy', 0.036749578960104896), ('sensor', 0.03643239059422336), ('standard', 0.03629971050484718), ('making', 0.036140448405663564), ('vision', 0.036114566826843775), ('framework', 0.03595701190011904), ('mechanism', 0.035773906888757034), ('logic', 0.03564596019631151), ('security', 0.03563192356110676), ('robotics',

0.03544707498868172), ('internet', 0.035018021606779845), ('natural', 0.03494224647693623), ('optimization', 0.034925374228958715), ('issue', 0.0348119537845443), ('detection', 0.034693265789291355), ('component', 0.03457382958136385), ('operation', 0.034513917456006966), ('classification', 0.03450668081146639), ('requirement', 0.03450046737552673), ('device', 0.03439177647083458), ('show', 0.0342035764077264), ('make', 0.03403902891672993), ('need', 0.03400068849048622), ('domain', 0.03387217986766891), ('function', 0.03363607578076371), ('general', 0.03360692711666859), ('including', 0.033598094121462835), ('vehicle', 0.03358709115268296), ('way', 0.03356719771351478), ('advanced', 0.033551938986721125), ('representation', 0.033495516297845376), ('current', 0.03346307154927041), ('three', 0.03344746768692419), ('dynamic', 0.033187012198096014), ('specific', 0.03306738821499363), ('action', 0.03289785610745299), ('task', 0.0327661403021676), ('ability', 0.032596995088768066), ('object', 0.0325684365593878), ('deep', 0.03222631715510408), ('order', 0.032198760647562656), ('aiestate', 0.03208213223435421), ('rule', 0.0317807381284339), ('potential', 0.031743191424535994), ('example', 0.03161537969054187), ('big', 0.0313642902725437), ('high', 0.031324509141270565), ('experience', 0.03112417753328443), ('mining', 0.030918573483629502), ('part', 0.030807387932270447), ('number', 0.03070280572306508), ('focus', 0.030636481140248712), ('modeling', 0.030628883082636747), ('may', 0.030608668551973566), ('goal', 0.030473165065529148), ('interface', 0.030457131737604403), ('structure', 0.030452803543683668), ('brain', 0.030431680177708156), ('student', 0.03042254324903523), ('pattern', 0.030258419390591648), ('methodology', 0.030201524412092277), ('programming', 0.029958576081430977), ('medical', 0.02994622273012678), ('several', 0.0297537406360227), ('designed', 0.029553594974902807), ('given', 0.029530655784321794), ('scheduling', 0.02952778308919023), ('interaction', 0.029471039030269878), ('efficient', 0.029095042987478843), ('autonomous', 0.028949060462650868), ('inference', 0.02878894677826514), ('real', 0.02878804862473605), ('accuracy', 0.028674757099303592), ('developing', 0.028627705152390232), ('education', 0.02849400142146857), ('automatic', 0.028489332716645204), ('article', 0.02817613175379202), ('related', 0.028145299253029332), ('equipment', 0.02814257862100065), ('challenge', 0.028060259032058422), ('quality', 0.028030506747937545), ('adaptive', 0.0278349775367457), ('state', 0.027824168960445827), ('form', 0.027695354842396696), ('energy', 0.027687434428360184), ('present', 0.027627325906518847), ('aim', 0.027592443420505014)]}

Results of Interesting areas of the lecturers of the Faculty of Information Technology, University of Moratuwa.

In here it displays the results according to the selected six areas of the lecturers of the faculty of Information Technology generated through the system developed.

- ✓ Dr. Thushari Silva

As sum of each array length into % [probability = True](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
20.0	30.0	30.0	10.0	10.0	0.0

As length of all abstracts into % [probability = True](output sum != 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
4.35	4.35	13.04	4.35	0.0	0.0

As length of all abstracts into % [probability = False](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
21.74	65.22	4.35	4.35	4.35	12.0

Using LSA (output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
13.04	43.47	0.0	21.73	17.39	0.0

- ✓ Dr. Upeksha Ganegoda

As sum of each array length into % [probability = True](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
7.69	7.69	30.77	0.0	53.85	0.0

As length of all abstracts into % [probability = True](output sum != 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
0.0	4.55	27.27	0.0	4.55	0.0

As length of all abstracts into % [probability = False](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
18.18	40.91	22.73	0.0	18.18	0.0

Using LSA (output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
13.63	31.81	0.0	27.27	18.18	0.0

✓ Professor Asoka Karunananda

As sum of each array length into % [probability = True](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
53.85	15.38	3.85	5.77	0.0	21.15

As length of all abstracts into % [probability = True](output sum != 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
28.0	8.0	2.0	3.0	0.0	11.0

As length of all abstracts into % [probability = False](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
41.0	39.0	2.0	6.0	0.0	12.0

Using LSA (output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
20.0	5.0	0.0	56.99	8.0	0.0

✓ Dr. Lochandaka Ranathunge

As sum of each array length into % [probability = True](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
31.58	36.84	21.05	10.53	0.0	0.0

As length of all abstracts into % [probability = True](output sum != 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
13.33	5.0	16.67	5.0	0.0	0.0

As length of all abstracts into % [probability = False](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
41.67	43.33	6.67	8.33	0.0	0.0

Using LSA (output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
18.33	18.33	0.0	58.33	1.66	0.0

✓ Dr. Supunmali Ahangama

As sum of each array length into % [probability = True](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
0.0	80.0	20.0	0.0	0.0	0.0

As length of all abstracts into % [probability = True](output sum != 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
0.0	0.0	18.75	0.0	0.0	0.0

As length of all abstracts into % [probability = False](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
6.25	87.5	6.25	0.0	0.0	0.0

Using LSA (output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
6.25	56.25	0.0	31.25	0.0	0.0

✓ Dr. Sagara Sumathipala

As sum of each array length into % [probability = True](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
50.0	37.5	0.0	0.0	6.25	6.25

As length of all abstracts into % [probability = True](output sum != 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
41.18	0.0	0.0	5.88	0.0	0.0

As length of all abstracts into % [probability = False](output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
47.06	41.18	0.0	0.0	5.88	5.88

Using LSA (output sum = 100)

ML	Big Data	Data Min:	Com: Vision	Bioinf:	AI
41.17	11.76	0.0	23.52	11.76	0.0