Project Report

on

Research Topic Recommendation System

Submitted in partial fulfillment for the award of
Post Graduate Diploma in Big Data Analytics
from **C-DAC ACTS (Pune)**

Guided by:
Mr. Prateek Maheshwari

Presented by:

Mr. Dhananjay Srivastava	**PRN: 190840125009**
Ms. Riddhi Tripathi	**PRN: 190840125048**
Mr. Samir Pokharkar	**PRN: 190840125032**
Mr. Sripad Pujari	**PRN: 190840125036**

Centre for Development of Advanced Computing (C-DAC), Pune

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Mr. Dhananjay Srivastava
Ms. Riddhi Tripathi
Mr. Samir Pokharkar
Mr. Sripad Pujari

have successfully completed their project on

Research Topic Recommendation System

under the guidance of Mr. Prateek Maheshwari

Mr. Prateek Maheshwari
(Project Guide)

(Project Supervisor)

Mr.Aditya Kumar Sinha
(Head of Department, ACTS)

ACKNOWLEDGEMENT

This project "**Research Topic Recommendation System** " was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We all are very glad to mention the name of *Mr.Prateek Maheshwari* for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

We are highly grateful to Ms. Risha P.R. (Manager (ACTS training Centre), C-DAC), for her guidance and support whenever necessary while doing this course Post Graduate Diploma in *Big Data Analytics(PGDBDA)* through C-DAC ACTS, Pune.

Our most heartfelt thanks goes to *Ms. Shilpi Shalini* (Course Coordinator, *PGDBDA*) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

From:
Mr. Dhananjay Srivastava (190840125009)
Ms. Riddhi Tripathi (190840125048)
Mr. Samir Pokharkar (190840125032)
Mr. Sripad Pujari (190840125036)

Abstract

With exponentially growing literature amounts, it has become increasingly impractical to create a complete literature review. This necessitates the need for automated search for relevant information with regards to modern research. The designed system is able to generate relevant and related topics from a given seed topic enabling the researcher to explore these in more detail. Moreover a functionality has been provided to automatically extract relevant keywords from the provided abstract. Finally, an internet based user interface is created for easy access and retrieval of information.

Table of Contents

1. Introduction and Overview of Project

Literature review is the systematic study of existing literature to properly understand the current research scenario before embarking on a research project. For Master's students reviewing existing literature takes up a exorbitant amount of time. Due to exponentially growing amount of literature it is becoming increasingly infeasible to conduct a completely manual literature review.

Several attempts have been made to address the problem of automating and systemizing the process of literature review recommendation. Several online tools such as SCImago, Scopus, Citenet Explorer, Web of Science Database, etc. which analyze citation network to give a comprehensive view of the current subject matter. Relatively unexplored domain is the area of topic recommendation which involves understanding the relevant sister topics to a main topic. This is the primary focus of our system.

The designed system is able to achieve this objective by analyzing the co-occurrence of topics in a large database to provide relevant recommendation. It can also recommend research topics on a submitted research paper abstract.

2. System Requirements

2.1 Dataset Description :

The data set is designed for research purpose only. The citation data is extracted from DBLP, ACM, MAG (Microsoft Academic Graph), and other sources.

In each text file, each line represents a paper, which is in JSON schema. The data schema is:

Field Name	Field Type	Description	Example
id	string	paper ID	53e9ab9eb7602d970354a97e
title	string	paper title	Data mining: concepts and techniques
authors.name	string	author name	Jiawei Han
author.org	string	author affiliation	Department of Computer Science, University of Illinois at Urbana-Champaign
author.id	string	author ID	53f42f36dabfaedce54dcd0c
venue.id	string	paper venue ID	53e17f5b20f7dfbc07e8ac6e
venue.raw	string	paper venue name	Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial
year	int	published year	2000
keywords	list of strings	keywords	["data mining", "structured data", "world wide web", "social network", "relational data"]
fos.name	string	paper fields of study	Web mining
fos.w	float	fields of study weight	0.659690857
references	list of strings	paper references	["4909282", "16018031", "16159250", "19838944", ...]
n_citation	int	citation number	40829
page_start	string	page start	11
page_end	string	page end	18
doc_type	string	paper type: journal, book title...	book
lang	string	detected language	en
publisher	string	publisher	Elsevier

Volume	string	volume	10
issue	string	issue	29
issn	string	issn	0020-7136
isbn	string	isbn	1-55860-489-8
doi	string	doi	10.4114/ia.v10i29.873
pdf	string	pdf URL	//static.aminer.org/upload/pdf/1254/370/239/53e9ab9eb7602d970354a97e.pdf
url	list	external links	["http://dx.doi.org/10.4114/ia.v10i29.873", "http://polar.lsi.uned.es/revista/index.php/ia/article/view/479"]
abstract	string	abstract	Our ability to generate...
indexed_abstract	dict	indexed abstract	{"IndexLength": 164, "InvertedIndex": {"Our": [0], "ability": [1], "to": [2, 7, ...]}}

2.2 Expected Input :

- Research Topic
- Research paper abstract

2.3 Software & Hardware :

- Windows 7 Workstation with 16GB of RAM
- DBLP-Citation-network V11 Dataset from Aminer.org
- MongoDB
- Python
- SciKit Learn
- MLxtend
- Flask

2.4 Assumptions & Constraints :

- The topic search is limited by the topics recorded within the particular database.
- A subsample of about 1,00,000 papers was taken from the original database to apply the keyword prediction from the abstract and FP-Growth based search due to practical time limit constraint.

3. System Workflow

3.1 Normal Search :

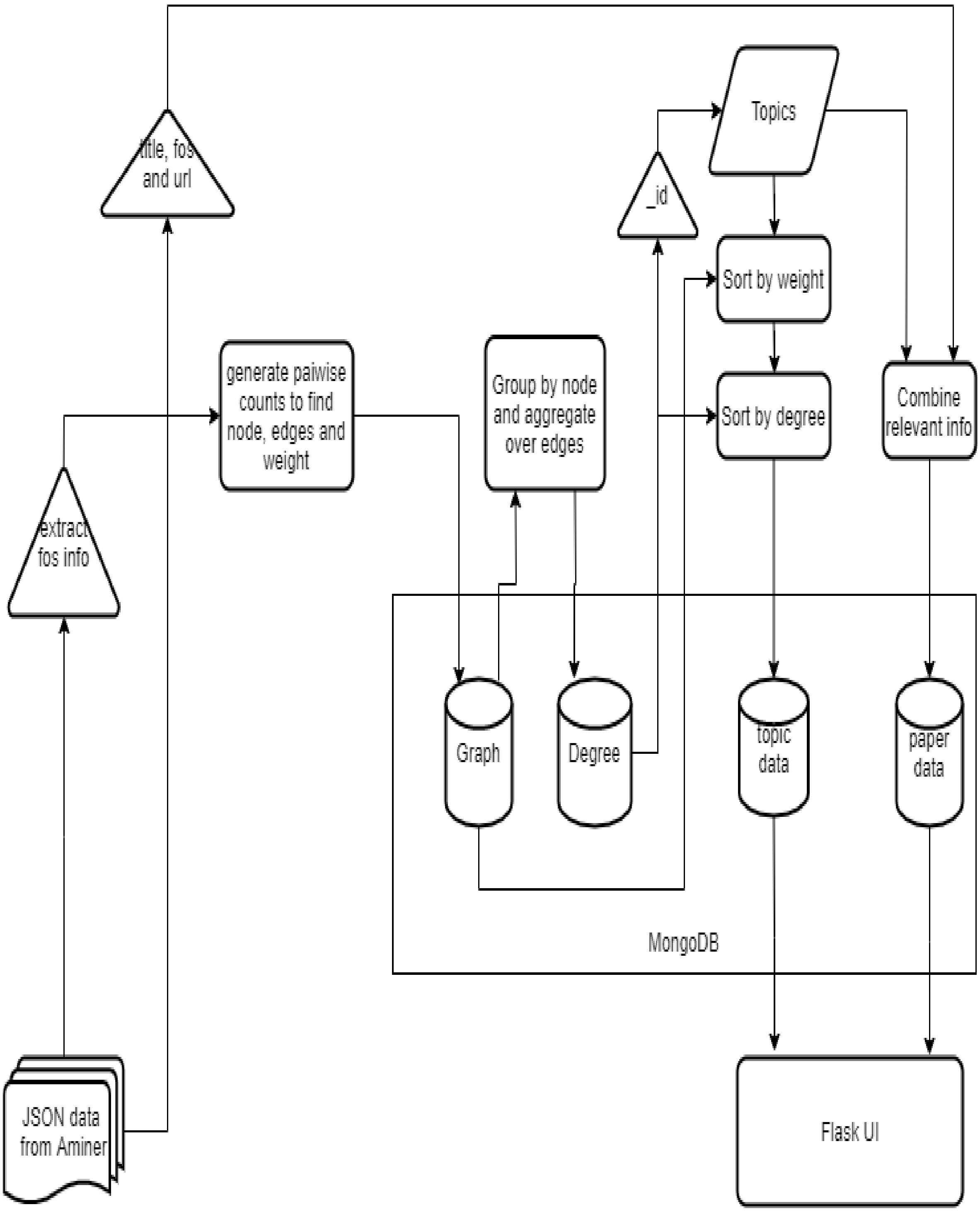


Fig 3.1 Normal Search

3.2 Advance Search :

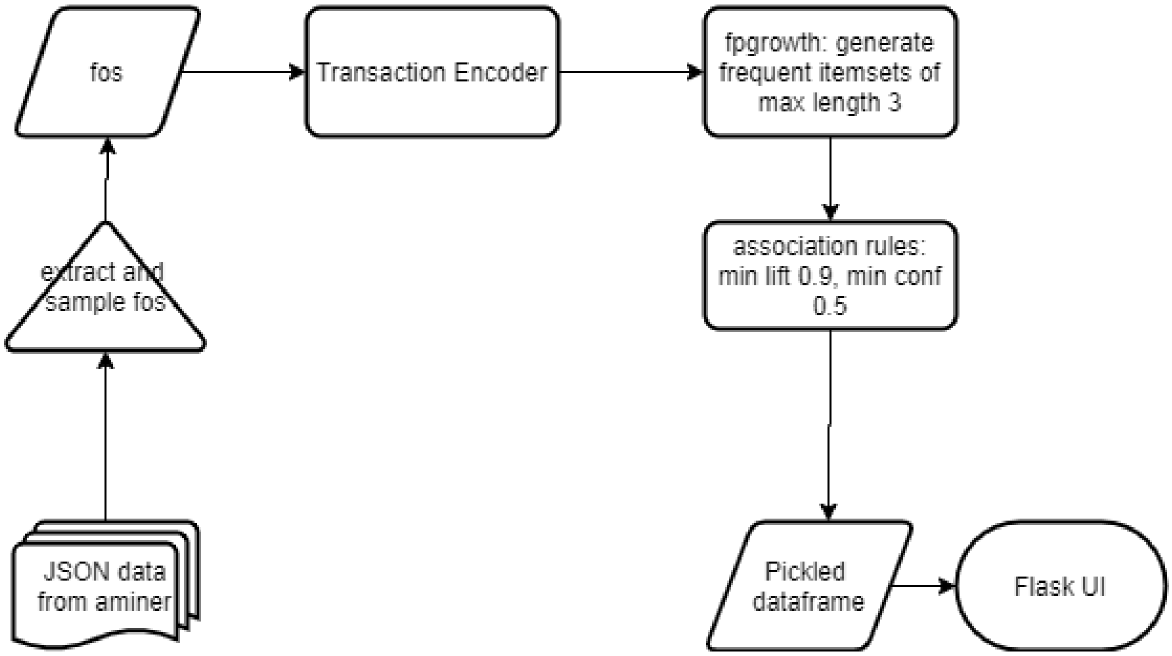


Fig 3.2 Advance Search

3.3 Abstract Based Topic Search :

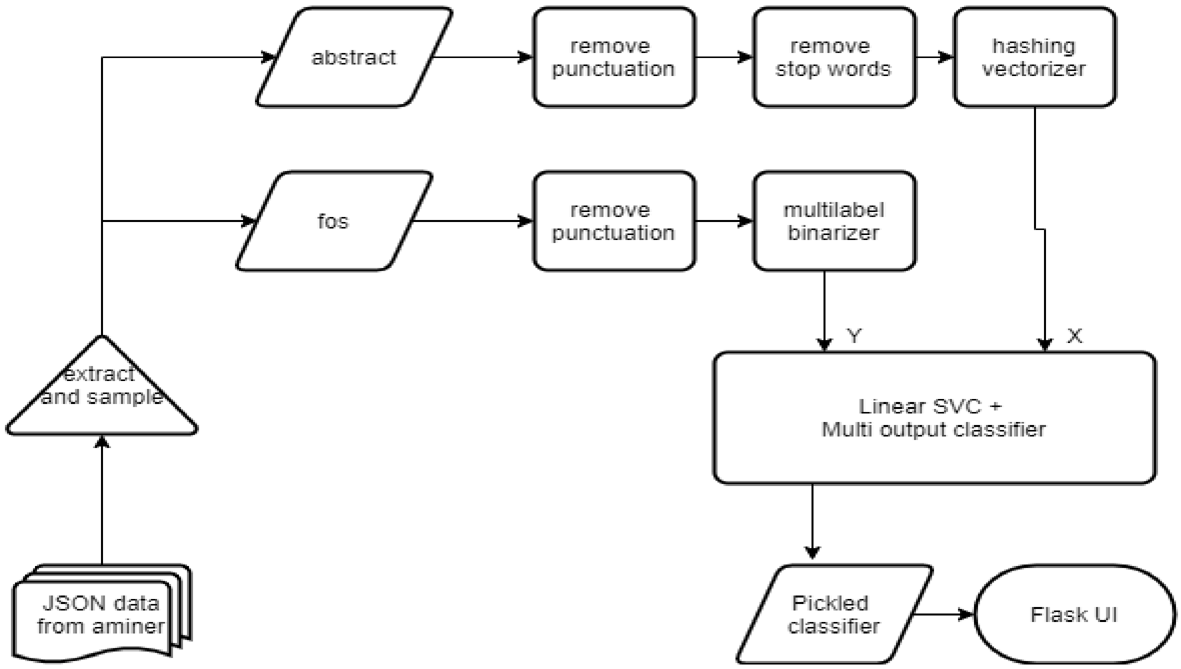


Fig 3.3 Abstract Based Topic Search

4. System Design

4.1 Data Preprocessing :

4.1.1 Preprocessing on database using Mongodb:

- Extracted relevant information primarily the fos, title, url and citation count.
- Performed computations using queries on MongoDB to create a collection named graph which consisted of node topic, related edge topics and it's weight.
- Performed grouping and summing aggregate operations to create degree collection.
- Sorted topics by weight and ranked by degree for all topics to create final collection topic_data used for normal search.
- Created another collection combining topics collection and extracted json data to create a collection paper_data searchable by topics.

4.1.2 Preprocessing on data for abstract based recommendation :

- Randomized 100000 sample of indexed_abstract and fos (field of study) is selected for keyword prediction based on the abstract.
- Regular expression is applied on the data to obtain each abstract as a list and remove all the characters other than A-Z or a-z and each word obtained is stored in a single row.
- Natural language processing NLTK library is used on that data to process the abstract.
- The preprocessing of the abstract data is done with the help of regular expression and removing stopping words with the help of nltk library.
- In the fos (field of study) column each field is stored in the form of list and is separated by comma using regular expression.
- **What are stop words ?**
- A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.
- Following is the code for removal of stop words from the data:

```
import nltk
from nltk.corpus import stopwords
set(stopwords.words(‘english’))
```

- On removing the stopping words from the abstract data will be formatted into bag of important words that can be further used for predicting keyword based on the abstract.
- After removing the stopping words the abstract data is further preprocessed to convert it into numerical values using sklearn.preprocessing library MultiLabelBinarizer.
- MultiLabelBinarizer encodes the column X into multiple binary columns per instance.
- Here MultiLabelBinarizer is used with the abstract data to convert each word in the paragraph of abstract as a columns or feature to fit the model.

4.1.3 Preprocessing on data for advance search using FP-Growth :

- In advance search fpgrowth algorithm is used for that the original data is sample into only fos data and further 100000 random samples are taken from it.

- To convert the categorical data to transaction encoder is used to convert data into Boolean true or false and sparse dataframe is used to store data to keep only the columns of fos that has true value.

4.2 Project Snapshots :

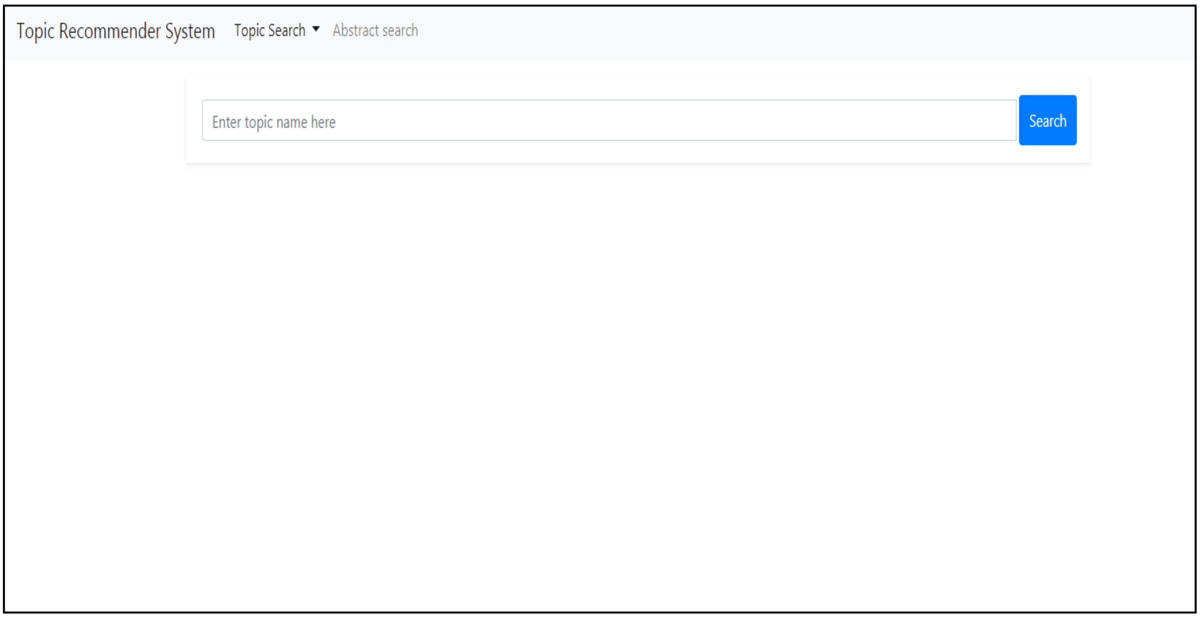


Fig 4.2.1 Main Layout

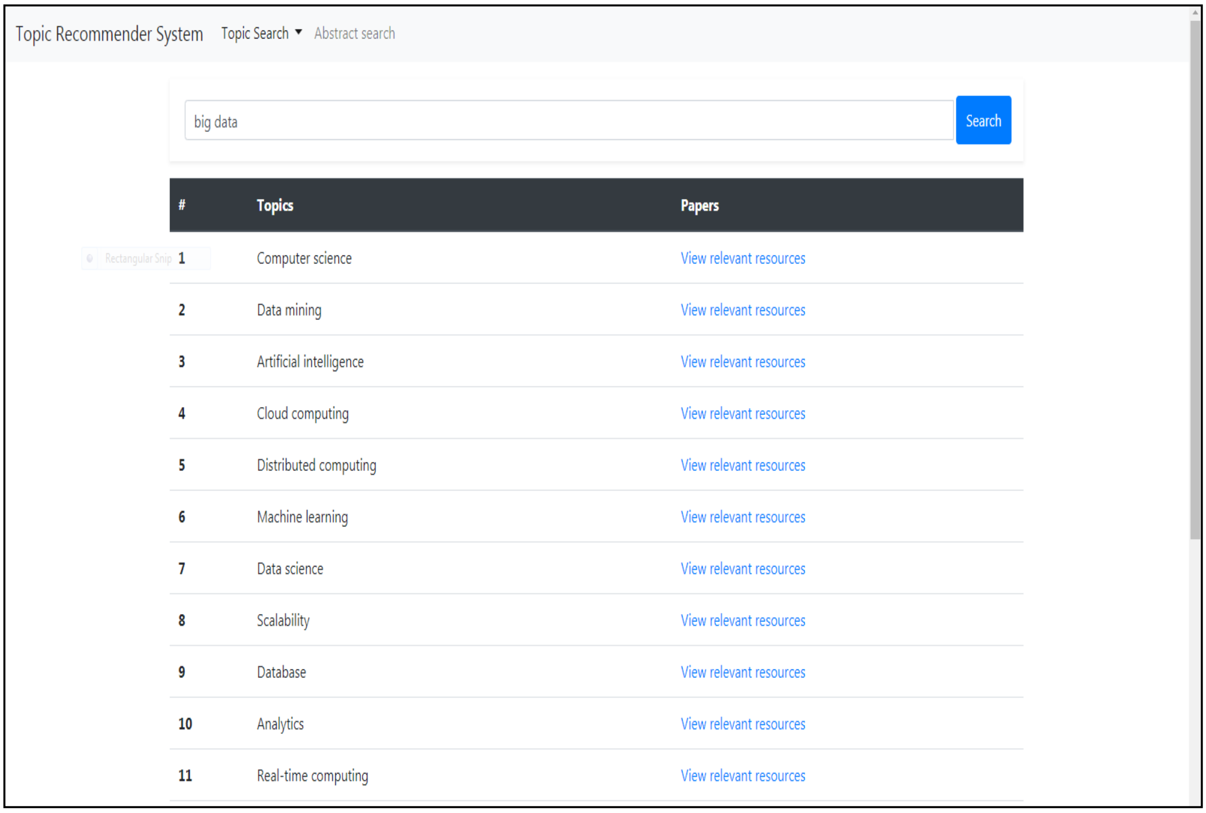
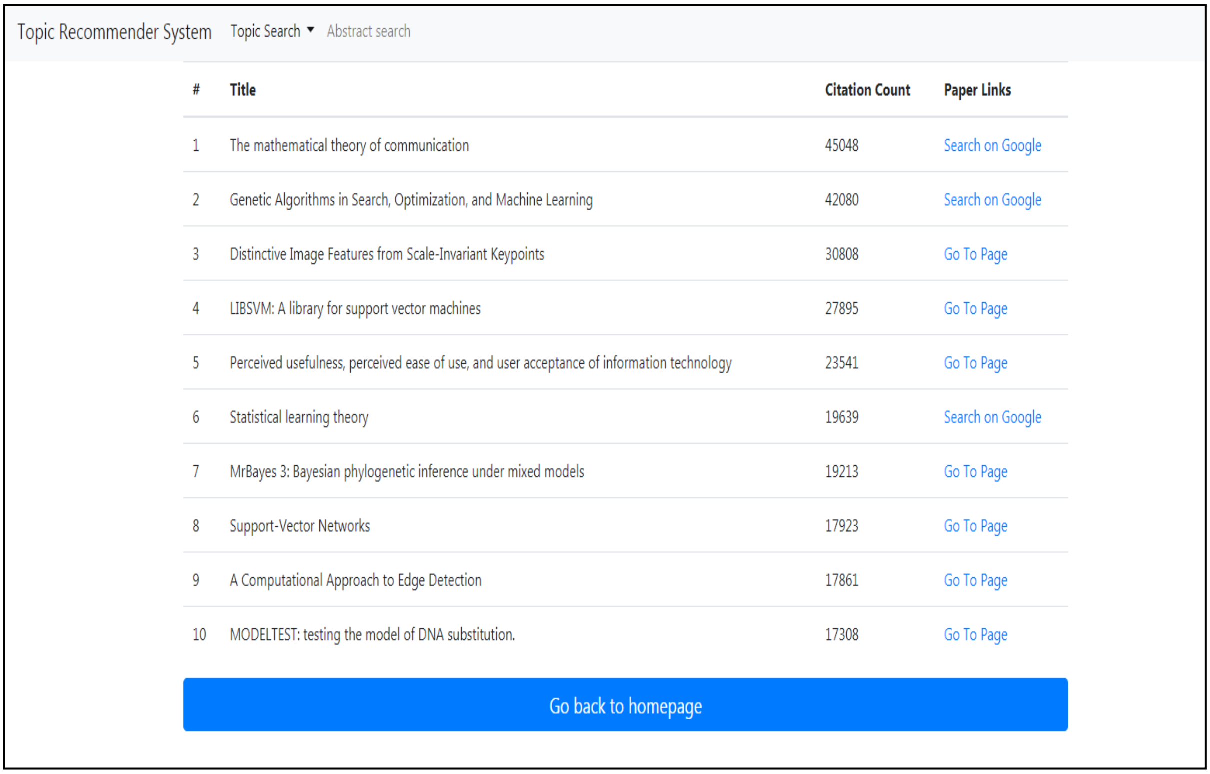


Fig 4.2.2 Normal Search

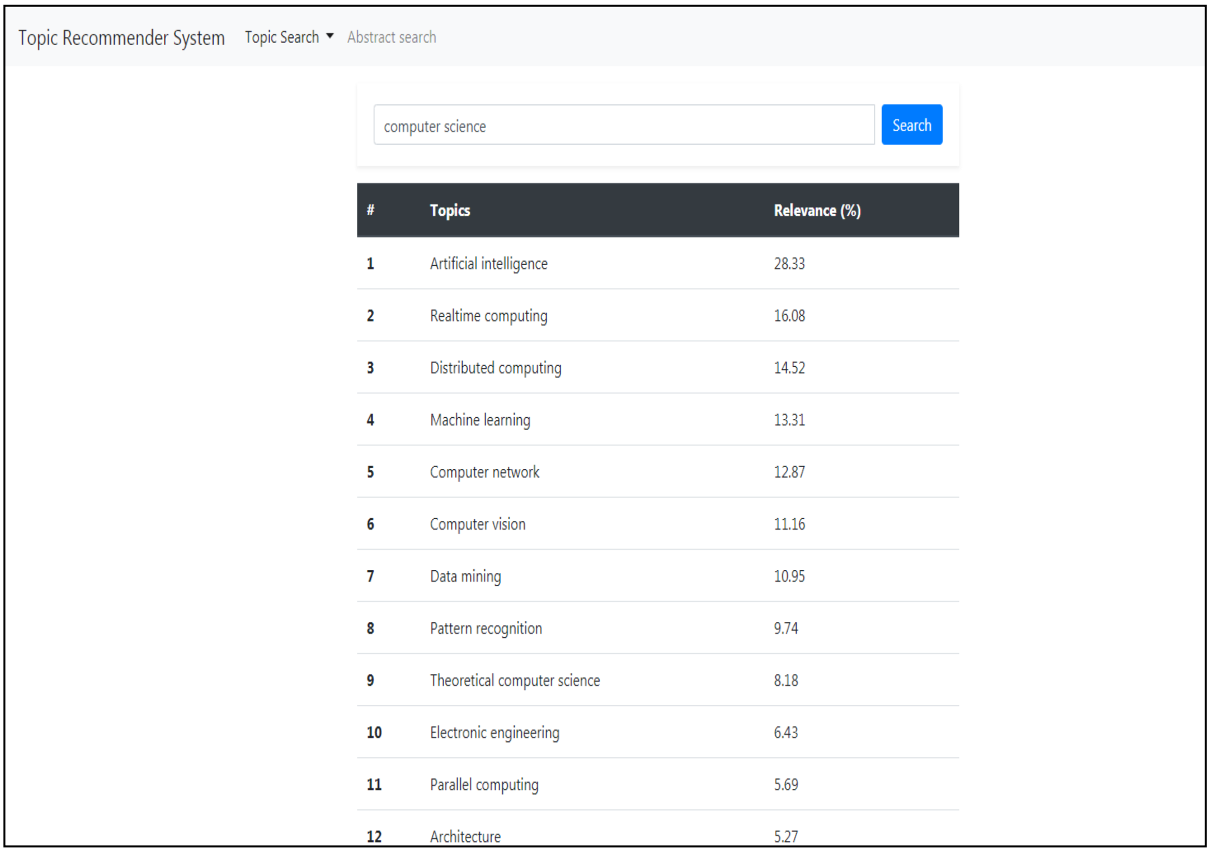


Topic Recommender System | Topic Search | Abstract search

#	Title	Citation Count	Paper Links
1	The mathematical theory of communication	45048	Search on Google
2	Genetic Algorithms in Search, Optimization, and Machine Learning	42080	Search on Google
3	Distinctive Image Features from Scale-Invariant Keypoints	30808	Go To Page
4	LIBSVM: A library for support vector machines	27895	Go To Page
5	Perceived usefulness, perceived ease of use, and user acceptance of information technology	23541	Go To Page
6	Statistical learning theory	19639	Search on Google
7	MrBayes 3: Bayesian phylogenetic inference under mixed models	19213	Go To Page
8	Support-Vector Networks	17923	Go To Page
9	A Computational Approach to Edge Detection	17861	Go To Page
10	MODELTEST: testing the model of DNA substitution.	17308	Go To Page

Go back to homepage

Fig 4.2.3 Paper Recommendation



Topic Recommender System | Topic Search | Abstract search

computer science | Search

#	Topics	Relevance (%)
1	Artificial intelligence	28.33
2	Realtime computing	16.08
3	Distributed computing	14.52
4	Machine learning	13.31
5	Computer network	12.87
6	Computer vision	11.16
7	Data mining	10.95
8	Pattern recognition	9.74
9	Theoretical computer science	8.18
10	Electronic engineering	6.43
11	Parallel computing	5.69
12	Architecture	5.27

Fig 4.2.4 Advance Search

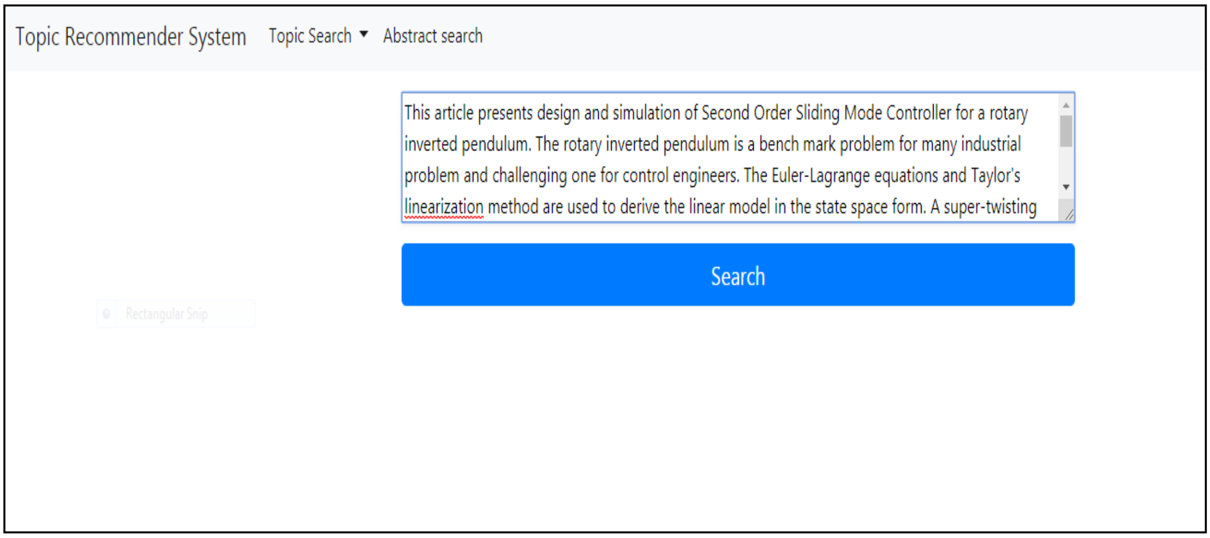


Fig 4.2.5 Abstract Based Topic Prediction

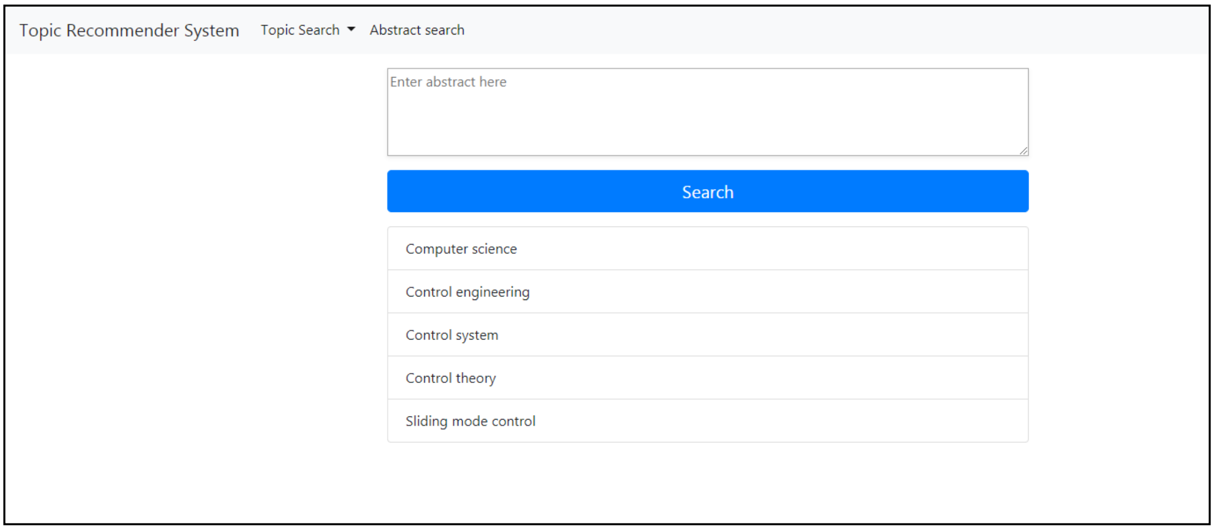


Fig 4.2.6 Abstract Based Topic Prediction

Fig 4.2.7 Fuzzy-Wuzzy Spelling Correction

5. System Implementation

5.1 Data extraction using Mongodb :

5.1.1 Aggregation :

Aggregation operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single purpose aggregation methods.

The most basic pipeline stages provide filters that operate like queries and document transformations that modify the form of the output document. Other pipeline operations provide tools for grouping and sorting documents by specific field or fields as well as tools for aggregating the contents of arrays, including arrays of documents. In addition, pipeline stages can use operators for tasks such as calculating the average or concatenating a string.

The pipeline provides efficient data aggregation using native operations within MongoDB, and is the preferred method for data aggregation in MongoDB. The aggregation pipeline can operate on a sharded collection. The aggregation pipeline can use indexes to improve its performance during some of its stages. In addition, the aggregation pipeline has an internal optimization phase.

5.1.2 Mongodb queries used for extracting data :

The image shows the sample of our raw data which we extracted from Aminer.org and imported into the MongoDB.

As we can see that there are multiple irrelevant fields in the sample data which we have to get rid to continue working with it. Hence to remove the unnecessary fields we executed the following query on the existing database.

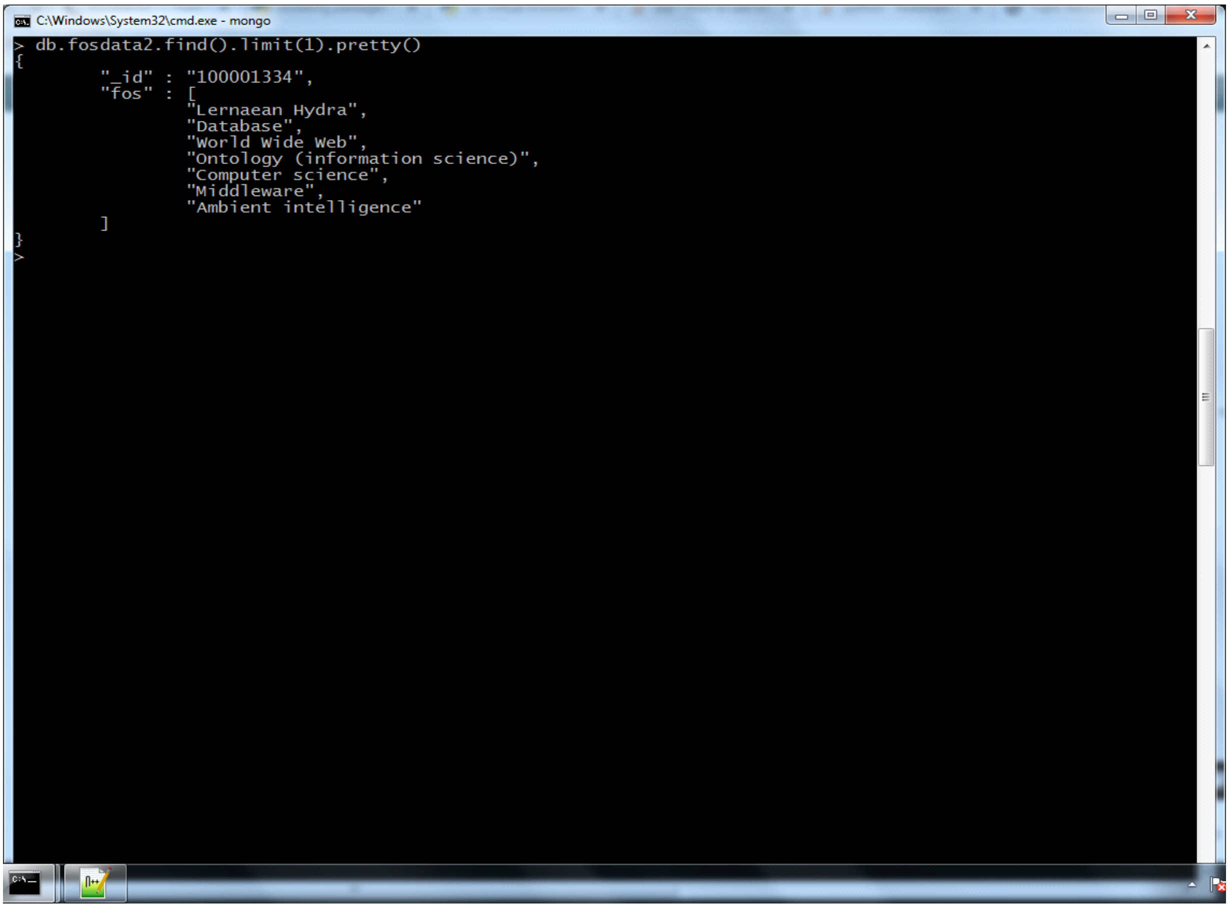
- **db.data.aggregate([{ $project : { "fos.name" : 1 , id : 1 , _id : 0} } , { $addFields : { fos : '$fos.name' } } , { $out : "fosdata" }]).pretty()**

So in the above query we extarcted the subfield "name" from main field "fos" into a new collection named "fosdata" using the aggregation framework of mongodb, fos represents field of study here which is the only required data as of now.

So as the resultant of above query we have all the names of fields of studies existing in the database into a single collection i.e. "fosdata".

C:\Windows\System32\cmd.exe - mongo

```
topics
> db.data.find().limit(1).pretty()
{
        "_id" : ObjectId("5de25f4e2b43ee98a35f7025"),
        "id" : "100001334",
        "title" : "Ontologies in HYDRA - Middleware for Ambient Intelligent Devices.",
        "authors" : [
                {
                        "name" : "Peter Kostelnik",
                        "id" : "2702511795"
                },
                {
                        "name" : "Martin Sarnovsky",
                        "id" : "2041014688"
                },
                {
                        "name" : "Jan Hreno",
                        "id" : "2398560122"
                }
        ],
        "venue" : {
                "raw" : "AMIF"
        },
        "year" : 2009,
        "n_citation" : 2,
        "page_start" : "43",
        "page_end" : "46",
        "doc_type" : "",
        "publisher" : "",
        "volume" : "",
        "issue" : "",
        "fos" : [
                {
                        "name" : "Lernaean Hydra",
                        "w" : 0.4178039
                },
                {
                        "name" : "Database",
                        "w" : 0.4269269
                },
                {
                        "name" : "World Wide Web",
                        "w" : 0.415332377
                },
                {
                        "name" : "Ontology (information science)",
                        "w" : 0.459045082
                },
                {
                        "name" : "Computer science",
                        "w" : 0.399807781
                },
                {
                        "name" : "Middleware",
                        "w" : 0.5905041
                },
                {
                        "name" : "Ambient intelligence",
                        "w" : 0.5440575
                }
        ]
}
>
```



We can see in the above image a sample of fosdata which we extracted from the last operation on the database. Now for further processing we required the total number of occurances of a pair of fields, it means that if a specific field of study is cited in another field of study then the count should be increased by one.

So for achieving this we executed the following query on the data.

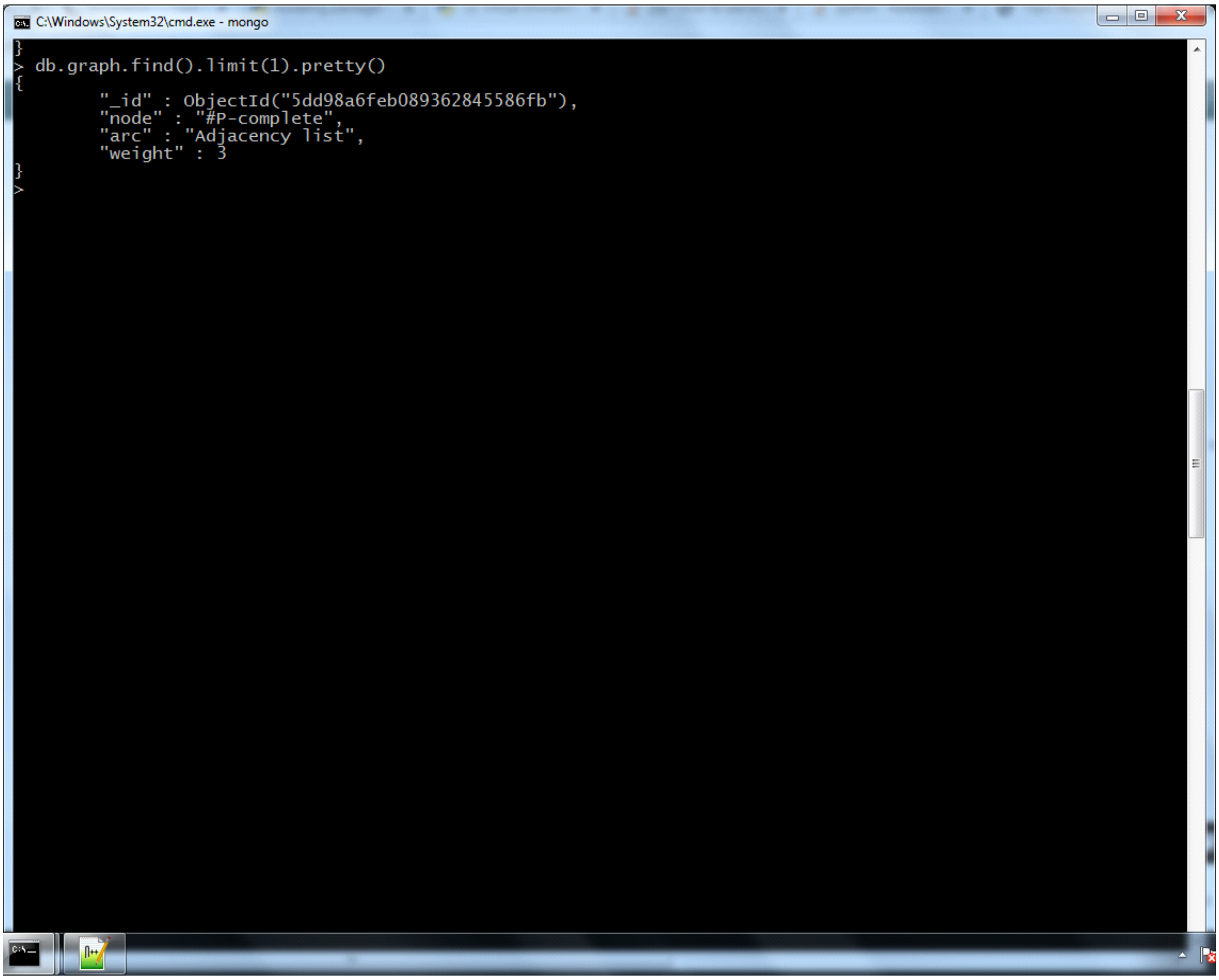
- db.fosdata2.aggregate([{

```
    $unwind : "$fos"
  }, {
    $lookup : {
      from : "fosdata2",
      localField : "_id",
      foreignField : "_id",
      as : "items"
    }
  }, {
    $unwind : "$items"
  }, {
    $unwind : "$items.fos"
  }, {
    $redact : {
      $cond : {
        if  : {
```

```
                $cmp : ["$fos", "$items.fos"]
              },
           then : "$$DESCEND",
           else  : "$$PRUNE"
        }
     }
  }, {
     $group : {
        _id : {
           k1 : "$fos",
           k2 : "$items.fos",
        },
        items : {
           $sum : 0.5
        }
     }
  }, {
     $sort : {
        "_id" : 1
     }
  }, {
     $project : {
        _id : 1,
        items : 1,
        a : {
           $cond : {
              if  : {
                 $eq : [{
                       $cmp : ["$_id.k1", "$_id.k2"]
                    }, 1]
              },
           then : "$_id.k2",
           else  : "$_id.k1"
        }
     },
     b : {
        $cond : {
           if  : {
              $eq : [{
                    $cmp : ["$_id.k1", "$_id.k2"]
                 }, -1]
           },
        then : "$_id.k2",
        else  : "$_id.k1"
     }
  },
}
}, {
```

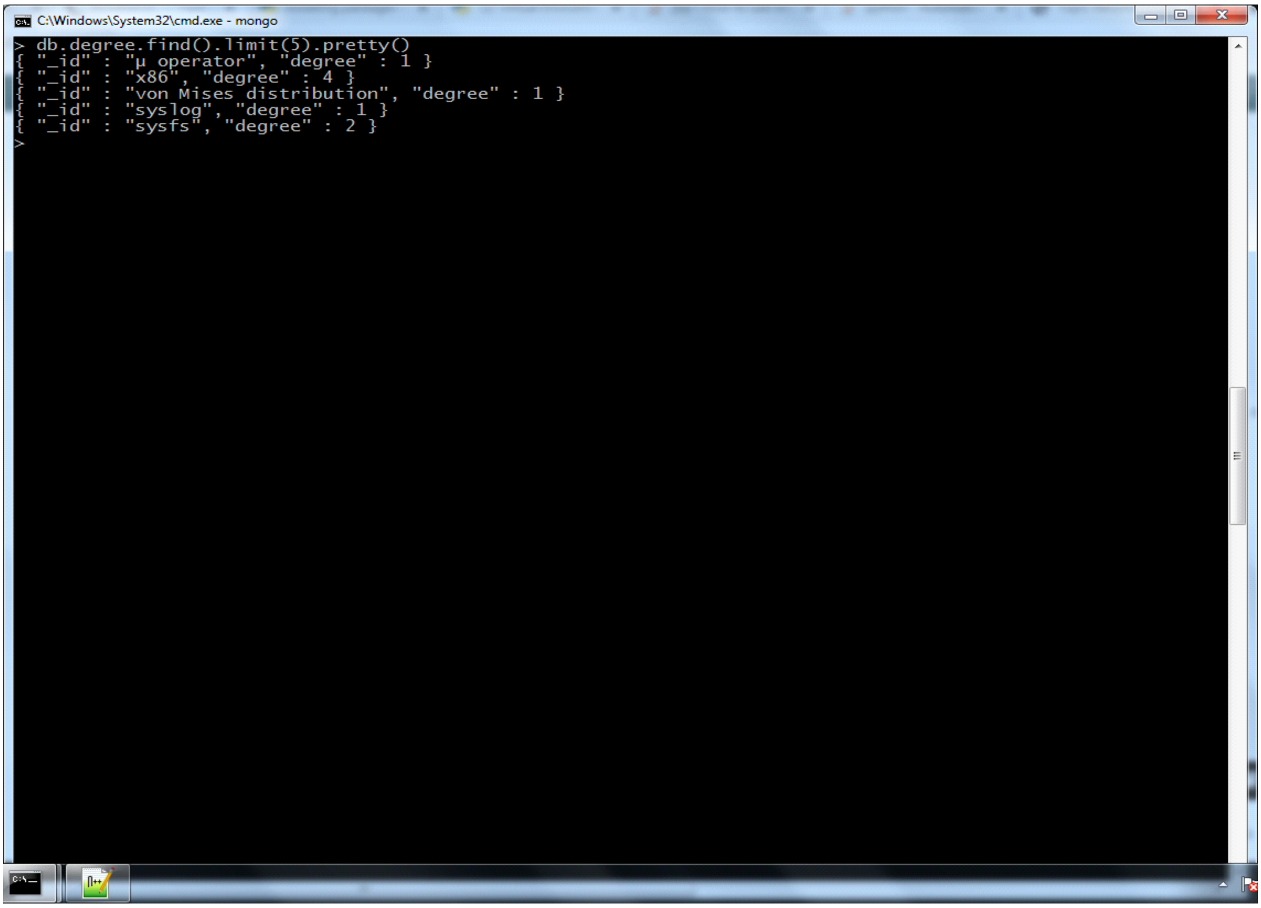
```
   $group : {
      _id : {
         k1 : "$a",
         k2 : "$b",
      },
      items : {
         $sum : "$items"
      }
   }
}, {
   $project : {
      _id : 0,
      node : "$_id.k1",
      arc : "$_id.k2",
      weight : "$items"
   }
} , { $out : "graph" }
] ,  {allowDiskUse : true})
```

The output of following query is shown in the following image. The node and arc fields are the names of fields of study.



The weight field in the above sample signifies only the number of occurrences of the fields but to understand the clustering properly we require an additional measure of degree of a field to decide the main fields from the set of fields. The logic follows here that a field with higher degree is the main field among the closely related fields this calculation considers the weight as well as degree of a field to decide the subfields and main fields. As a result we will get a cluster of fields of studies.

- db.graph.aggregate([{$group:{_id:"$node",degree:{$sum: 1}}},{$out:"degree"}]).pretty()



- db.data.aggregate([{$match:{$and:[{'indexed_abstract.InvertedIndex':{$ne:null}},{fos:{$ne:null}},
  {n_citation:{$ne:null}},{doi:{$ne:null}}]}},{$sample:{size:100000}},{$project:{'_id':'$id',

```
                                                     'title':1,
                                                     'authors':1,
                                                     'venue.raw':1,
                                                     'year':1,
                                                     'n_citation':1,
                                   'indexed_abstract.InvertedIndex':1,
                                                     'fos': 1,
                                                     'doi':1,}},
{$out:"final_data"}],{allowDiskUse: true})
```

Randomly selects 1,00,000 sample from the whole data. This selective sample is required for the relevant topic recommendation based on the given abstract using text mining.

5.2 Algorithm Implementation Using Python3

5.2.1 PyMongo :

The PyMongo distribution contains tools for interacting with MongoDB database from Python. The bson package is an implementation of the BSON format for Python.

The PyMongo package is a native Python driver for MongoDB. The gridfs package is a gridfs implementation on top of PyMongo.

As mentioned above we also used PyMongo for interaction with our database to extract data from database and perform python scripting on it for functioning and calculations.

5.2.2 Pickle :

Pickle — Python object serialization :

The pickle module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” or “flattening”.

5.2.3 Fuzzy-Wuzzy :

There are many methods of comparing string in python. Some of the main methods are:

1. Using regex
2. Simple compare
3. Using difflib

But one of the very easy method is by using fuzzy-wuzzy library where we can have a score out of 100, that denotes two string are equal by giving similarity index.

We used this library of Python which is used for string matching. Fuzzy string matching is the process of finding strings that match a given pattern. Basically it uses Levenshtein Distance to calculate the differences between sequences.

5.2.4 SciKit Learn :

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

The library is focused on modeling data. It is not focused on loading, manipulating and summarizing data. For these features, refer to NumPy and Pandas. We use the SciKit-learn for data preprocessing.

5.2.4.1 Hashing Vectorizer :

It turns a collection of text documents into a scipy.sparse matrix holding token occurrence counts (or binary occurrence information), possibly normalized as token frequencies if norm=’l1’ or projected on the Euclidean unit sphere if norm=’l2’.

This text vectorizer implementation uses the hashing trick to find the token string name to feature integer index mapping.

- **This strategy has several advantages :**

1. It is very low memory scalable to large datasets as there is no need to store a vocabulary dictionary in memory

2. It is fast to pickle and un-pickle as it holds no state besides the constructor parameters

3. It can be used in a streaming (partial fit) or parallel pipeline as there is no state computed during fit.

5.2.4.2 Multi Label Binarizer :

Binarize labels in a one-vs-all fashion. Several regression and binary classification algorithms are available in scikit-learn. A simple way to extend these algorithms to the multi-class classification case is to use the so-called one-vs-all scheme.

At learning time, this simply consists in learning one regressor or binary classifier per class. In doing so, one needs to convert multi-class labels to binary labels (belong or does not belong to the class). Label Binarizer makes this process easy with the transform method.

At prediction time, one assigns the class for which the corresponding model gave the greatest confidence. Label Binarizer makes this easy with the inverse transform method.

5.2.4.3 Multi Output Classifier :

This strategy consists of fitting one classifier per target. This is a simple strategy for extending classifiers that do not natively support multi-target classification.

5.2.4.4 Linear SVC :

Linear Support Vector Classification. Similar to SVC with parameter kernel=’linear’, but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.

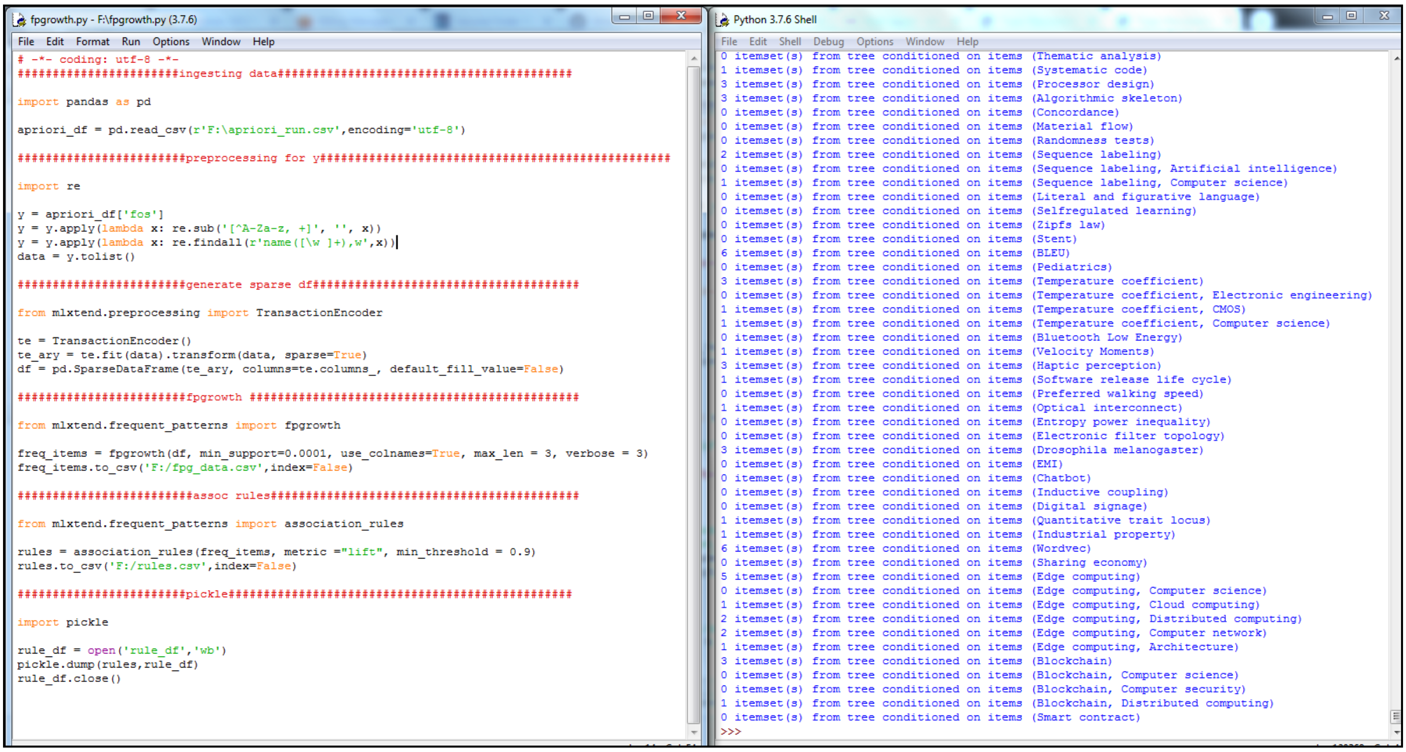
5.2.5 MLxtend :

5.2.5.1 FP Growth :

FP-Growth is an algorithm for extracting frequent itemsets with applications in association rule learning that emerged as a popular alternative to the established Apriori algorighm.

In general, the algorithm has been designed to operate on databases containing transactions, such as purchases by customers of a store. An itemset is considered as "frequent" if it meets a user-specified support threshold. For instance, if the support threshold is set to 0.5 (50%), a frequent itemset is defined as a set of items that occur together in at least 50% of all transactions in the database.

In particular, and what makes it different from the Apriori frequent pattern mining algorithm, FP-Growth is an frequent pattern mining algorithm that does not require candidate generation. Internally, it uses a so-called FP-tree (frequent pattern tree) data structure without generating the candidate sets explicitly, which makes is particularly attractive for large datasets.




```
# -*- coding: utf-8 -*-
#########################ingesting data##########################################

import pandas as pd

apriori_df = pd.read_csv(r'F:\apriori_run.csv',encoding='utf-8')

#########################preprocessing for y#################################################

import re

y = apriori_df['fos']
y = y.apply(lambda x: re.sub('[^A-Za-z, +]', '', x))
y = y.apply(lambda x: re.findall(r'name([\w ]+),w',x))
data = y.tolist()

#########################generate sparse df########################################

from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(data).transform(data, sparse=True)
df = pd.SparseDataFrame(te_ary, columns=te.columns_, default_fill_value=False)

#########################fpgrowth ###############################################

from mlxtend.frequent_patterns import fpgrowth

freq_items = fpgrowth(df, min_support=0.0001, use_colnames=True, max_len = 3, verbose = 3)
freq_items.to_csv('F:/fpg_data.csv',index=False)

##########################assoc rules############################################

from mlxtend.frequent_patterns import association_rules

rules = association_rules(freq_items, metric ="lift", min_threshold = 0.9)
rules.to_csv('F:/rules.csv',index=False)

#########################pickle##################################################

import pickle

rule_df = open('rule_df','wb')
pickle.dump(rules,rule_df)
rule_df.close()
```

```
0 itemset(s) from tree conditioned on items (Thematic analysis)
1 itemset(s) from tree conditioned on items (Systematic code)
3 itemset(s) from tree conditioned on items (Processor design)
3 itemset(s) from tree conditioned on items (Algorithmic skeleton)
0 itemset(s) from tree conditioned on items (Concordance)
0 itemset(s) from tree conditioned on items (Material flow)
0 itemset(s) from tree conditioned on items (Randomness tests)
2 itemset(s) from tree conditioned on items (Sequence labeling)
0 itemset(s) from tree conditioned on items (Sequence labeling, Artificial intelligence)
1 itemset(s) from tree conditioned on items (Sequence labeling, Computer science)
0 itemset(s) from tree conditioned on items (Literal and figurative language)
0 itemset(s) from tree conditioned on items (Selfregulated learning)
0 itemset(s) from tree conditioned on items (Zipfs law)
0 itemset(s) from tree conditioned on items (Stent)
6 itemset(s) from tree conditioned on items (BLEU)
0 itemset(s) from tree conditioned on items (Pediatrics)
3 itemset(s) from tree conditioned on items (Temperature coefficient)
0 itemset(s) from tree conditioned on items (Temperature coefficient, Electronic engineering)
1 itemset(s) from tree conditioned on items (Temperature coefficient, CMOS)
1 itemset(s) from tree conditioned on items (Temperature coefficient, Computer science)
0 itemset(s) from tree conditioned on items (Bluetooth Low Energy)
1 itemset(s) from tree conditioned on items (Velocity Moments)
3 itemset(s) from tree conditioned on items (Haptic perception)
1 itemset(s) from tree conditioned on items (Software release life cycle)
0 itemset(s) from tree conditioned on items (Preferred walking speed)
1 itemset(s) from tree conditioned on items (Optical interconnect)
0 itemset(s) from tree conditioned on items (Entropy power inequality)
0 itemset(s) from tree conditioned on items (Electronic filter topology)
3 itemset(s) from tree conditioned on items (Drosophila melanogaster)
0 itemset(s) from tree conditioned on items (EMI)
0 itemset(s) from tree conditioned on items (Chatbot)
0 itemset(s) from tree conditioned on items (Inductive coupling)
0 itemset(s) from tree conditioned on items (Digital signage)
1 itemset(s) from tree conditioned on items (Quantitative trait locus)
1 itemset(s) from tree conditioned on items (Industrial property)
6 itemset(s) from tree conditioned on items (Wordvec)
0 itemset(s) from tree conditioned on items (Sharing economy)
5 itemset(s) from tree conditioned on items (Edge computing)
0 itemset(s) from tree conditioned on items (Edge computing, Computer science)
1 itemset(s) from tree conditioned on items (Edge computing, Cloud computing)
2 itemset(s) from tree conditioned on items (Edge computing, Distributed computing)
2 itemset(s) from tree conditioned on items (Edge computing, Computer network)
1 itemset(s) from tree conditioned on items (Edge computing, Architecture)
3 itemset(s) from tree conditioned on items (Blockchain)
0 itemset(s) from tree conditioned on items (Blockchain, Computer science)
0 itemset(s) from tree conditioned on items (Blockchain, Computer security)
1 itemset(s) from tree conditioned on items (Blockchain, Distributed computing)
0 itemset(s) from tree conditioned on items (Smart contract)
>>>
```

5.2.5.2 NLP :

NLP is used to analyze text, allowing machines to understand how human's speak. This human-computer interaction enables real-world applications like automatic text summarization, sentiment analysis, topic extraction, named entity recognition, parts-of-speech tagging, relationship extraction, stemming, and more. NLP is commonly used for text mining, machine translation, and automated question answering.

5.2.5.2.1 NLTK :

Natural Language Toolkit (NLTK): a Python library that provides modules for processing text, classifying, tokenizing, stemming, tagging, parsing, and more.

Natural Language Processing with PythonNatural language processing (nlp) is a research field that presents many challenges such as natural language understanding.

Text may contain stop words like ‘the’, ‘is’, ‘are’. Stop words can be filtered from the text to be processed. There is no universal list of stop words in nlp research, however the nltk module contains a list of stop words.

5.2.6 Flask :

Flask is a lightweight WSGI (Web Server Gateway Interface) web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around 'Werkzeug' (comprehensive WSGI web application library) and 'Jinja' and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

preprocessing.py - F:\preprocessing.py (3.7.6)

```
from sklearn.svm import LinearSVC
from sklearn.multioutput import MultiOutputClassifier

clf = MultiOutputClassifier(LinearSVC(),n_jobs=7)
clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)

y_pred_df = pd.DataFrame(y_pred,columns=y_test.columns)

#####################accuracy score##############################################

from sklearn.metrics import accuracy_score

acc = []

for i in range(len(in_vals)):
    val = accuracy_score(y_test.iloc[:,i],y_pred_df.iloc[:,i])
    #print(i,val)
    acc.append(val)
```

Python 3.7.6 Shell

```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========================= RESTART: F:\preprocessing.py =========================
mean accuracy is 0.9977200863930885
>>>
```

5.2.6.1 Jinja Syntax in Flask :

Jinja is one of the most used template engines for Python. It is inspired by Django's templating system but extends it with an expressive language that gives template authors a more powerful set of tools. On top of that it adds sandboxed execution and optional automatic escaping for applications where security is important.

It is internally based on Unicode and runs on a wide range of Python versions.

5.2.6.2 Method 'app.route' in Flask :

Modern web frameworks use the routing technique to help a user remember application URLs. It is useful to access the desired page directly without having to navigate from the home page.

The route() decorator in Flask is used to bind URL to a function. For example ,

```
@app.route(‘/hello’)
def hello_world():
   return ‘hello world’
```

Here, URL ‘/hello’ rule is bound to the hello_world() function. As a result, if a user visits http://localhost:5000/hello URL, the output of the hello_world() function will be rendered in the browser.

5.2.6.3 Request Handling in Flask using Request Object :

The data from a client’s web page is sent to the server as a global request object. In order to process the request data, it should be imported from the Flask module.

Important attributes of request object are listed below :

- **Form** − It is a dictionary object containing key and value pairs of form parameters and their values.
- **args** − parsed contents of query string which is part of URL after question mark (?).
- **files** − data pertaining to uploaded file.
- **method** − current request method.

5.2.6.4 Template Rendering in Flask :

It is possible to return the output of a function bound to a certain URL in the form of HTML. For instance, in the following script, hello() function will render ‘Hello World’ with <h1> tag attached to it.

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def index():
   return '<html><body><h1>Hello World</h1></body></html>'
if __name__ == '__main__':
   app.run(debug = True)
```

However, generating HTML content from Python code is cumbersome, especially when variable data and Python language elements like conditionals or loops need to be put. This would require frequent escaping from HTML.

This is where one can take advantage of Jinja2 template engine, on which Flask is based. Instead of returning hardcode HTML from the function, a HTML file can be rendered by the render_template() function.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
   return render_template('hello.html')

if __name__ == '__main__':
   app.run(debug = True)
```

Flask will try to find the HTML file in the templates folder, in the same folder in which this script is present.

- Application folder
  - Hello.py
  - templates
    - hello.html

The term 'web templating system' refers to designing an HTML script in which the variable data can be inserted dynamically. A web template system comprises of a template engine, some kind of data source and a template processor.

6. Limitation And Future Enhancement

1. For now topics are being recommended based on the present limited dataset only.

2. Due to resource constraints the data samples used to topic prediction based on abstract and advance search are randomized 100000 samples.

3. Full dataset can be utilized for better algorithm accuracy and prediction and can be deployed on a scalable system.

4. Real time data for research topic recommendation can be taken using web scrapping related website.

5. Advanced algorithm can be used for recommending most related topics such as page rank algorithm.

Conclusion

Thus we can say that, the designed system fulfills the objective of research topic recommendation by involving multiple strategies like node degree and edge weight or the FP-Growth algorithm. Another tool provided to recommend topics based on abstract data is also included which has a good accuracy. Although there are multiple avenues for improvement such as increasing data size, query coverage, more structured web app, etc we believe that this system will be able to provide good value for it's users.

Links and References

- https://github.com/dhananjay-srivastava/cdac_final_project
- https://www.aminer.cn/citation
- https://clarivate.com/webofsciencegroup/solutions/web-of-science/
- https://www.citnetexplorer.nl/
- https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/
- https://en.wikipedia.org/wiki/Centrality
- https://en.wikipedia.org/wiki/Association_rule_learning#FP-growth_algorithm
- https://scikit-learn.org/stable/
- http://rasbt.github.io/mlxtend/
- https://www.palletsprojects.com/p/flask/