Research Paper Recommendation System

Project Group 1

Abhinav Balasubramanian
<abhinav.balasubramanian@sjsu.edu>
Student Id: 013853281
Department of Computer Engineering
San Jose State University
San Jose, CA

Naga Janaki Dwadasi
<nagajanaki.dwadasi@sjsu.edu>
Student Id: 013853034
Department of Computer Engineering
San Jose State University
San Jose, CA

Akshata Kulkarni
<Akshata.Kulkarni@sjsu.edu>
Student Id: 013829010
Department of Computer Engineering
San Jose State University
San Jose, CA

Manish Katturu
<manish.katturu@sjsu.edu>
Student Id: 013825201
Department of Computer Engineering
San Jose State University
San Jose, CA

https://github.com/akshatakulkarni98/Paper_Recommendation_system/

TABLE OF CONTENTS

1. ABSTRACT

In the modern days, with the increase in the number of technologies and number of research being conducted in each field, it becomes difficult in finding the research paper most relevant to paper of our interest. Since a lot of research papers are being published in the current time, it also becomes increasing difficult to validate the authenticity of the research papers published. For further performing an in-depth research on a particular topic, it becomes essential to track and study the research papers that has cited and has been cited by the research paper in hands. The main aim of this project is to build a hybrid content and collaborative based recommendation system that recommends the best suitable research paper to the researchers on the basis of popularity metrics, content similarity and the collaborative nature of citations and references. The system implemented is also deployed in google cloud to facilitate user interface that could be used by the visitors to find relevant research paper easily. The system is also evaluated by comparing its performance with the recommender system of the webpage from which the data scraped and is found to be performing much more efficiently than the existing recommender system.

2. INTRODUCTION

As a new medium of digital library, Research paper publishing platform provides an easy access to review papers online. In many universities and research institutions, students, professors, and other researchers search for the research papers related to their work. As a result, looking for the right papers has become a difficult part of their work. These paper publishing platforms provide common search results to all the users on a particular search. They lack in providing dynamic search results analyzing the preferences of the user, due to which the most popular papers which have been antiquated are appearing on the top of the search results leaving no scope to newly published papers. A research paper recommender system will benefit these people in helping them to find the most relevant papers and in saving their precious time. Many organizations like Amazon, Netflix etc. recommend commodities to users by analyzing customers preferences thereby improving the user experience. So, in order to improve the search results and provide a better paper recommendation based on the user's interest model, we present a personalized recommendation system which is effective in retrieving the most suitable papers to a user by analyzing the user online behavior and preferences.

Personalized recommendation systems mainly are of three categories, the rule-based filtering, the content-based filtering and the collaborative filtering. In rule-based filtering users provide their interest information, build and maintain their interest models by themselves. Hence, the scalability of such systems will be poor as the users take the responsibility for modeling. The collaborative filtering systems have been very successful in the past but some issues such as data sparse and scalability, have been revealed in their applications. The growth of the number of users and items will lead to the exponential computational complexity of such systems. In case that there is little information on the user's interest, the system may be unable to make any item recommendations for a particular user, which is the so-called "cold start" problem. The key of the content-based filtering system is how to construct the interesting model based on the user history collected automatically by the user.

In our recommendation system, we have collected various features of the research papers like paper titles, keywords, abstracts, number of citations, number of influenced papers etc. to build our dataset. We represent each research paper by a Term Frequency-Inverse Document Frequency(TF-IDF) vector calculating the TF-IDF scores for each word in the document and for all the research papers. Also, we collect the user search keywords, user click history to build the user interest model. We find the most similar documents to user interest model based on different metrics and rank those papers based on the weighted average of different features of the paper like popularity score, citation score, similarity score etc. to recommend papers to the users.

3. LITERATURE SURVEY

This section summarizes the various ideas and approaches that can be used for a research paper recommendation system. The approaches mostly used are content-based recommendations and collaborative filtering recommendations. Content-based recommendation model and collaborative-filtering recommendation model both work best with user data. User data helps to model their preferences and ultimately provide them with a customised experience. To fetch user data, ratings or interests of user can be fetched either explicit or implicit. However, acquiring user profile and interests along with research papers corpus for a research paper recommendation system is difficult as most research paper websites allow access to the documents without the need of a user profile in them. Content-based recommendation could still be performed by popular methods like TF-IDF(term frequency - inverse document frequency) which help calculate the frequency of a word in a document from a corpus. This method works best with fine tuned searches by user, as this method doesn't use any user preference. Collaborative-Filtering is majorly dependent on user based filtering or item based filtering which are done based on user ratings to items. [1] This paper by leveraging the advantages of collaborative filtering approach, presents a way to utilize the publicly available contextual metadata to infer the hidden associations that exist between research papers in order to personalize recommendations. The algorithm used for Collaborative Filtering is from [1] which is shown below:

Algorithm: Collaborative Research Paper Recommendation
Input: Target Paper
Output: Top-N Recommendation
Given a target paper p_i as a query,

1. Retrieve all the set of references Rf_j of the target paper pi from the paper-citation relation matrix C.
    a. For each of the referencesRf_j, extract all other papers pci that also cited Rfj other than the target paper p_i.
2. Retrieve all the set of citations Cf_j of the target paper pi from the paper-citation relation matrix C.
    a. For each of the citations Cf_j, extract all other papers pri that Cf_j referenced other than the target paper p_i.
3. Qualify all the candidate papers p_c from p_{ci} that has been referenced by at least any of the p_{ri}
4. Measure the extent of similarity $W^{pi->pc}$ between the target paper pi and the qualified candidate papers p_c
5. Recommend the top-N most similar papers to the user.

4. DATA COLLECTION

In this section, we will see how the data has been extracted from 'Semantic Scholar' website. We have chosen Semantic Scholar because it has a huge variety of data with respect to research papers, namely:

- Article Information
- Popularity score
- References and Citations Information

The columns present in the dataset and the information provided by them is provided in the table below

TABLE-1
Various Column information in the dataset used

Column Name	Information
Title of the Article	Title of the research article is provided
Authors	The names of the authors is provided
Published Journal	The name of the journal under which the article is published is provided
Year of Publication	Year in which the article is released is provided
Abstract of the article	This section contains a small summary about what is present in the article
Number of Citations	The number of times the given article is cited is provided

Column Name	Information
Highly Influenced papers	The count of research papers which uses the given research article as its main source of foundation
Cite Background	The count of research papers in which the background work of the given research article is cited
Cite Methods	The count of research papers in which the method employed by the given research article is cited

Cite Results	The count of research papers in which the final results of the given research article is cited
Twitter Mentions	The number of times the given article is tweeted on twitter
Citation Titles	Contains the name of articles that cites that particular paper
Citation Journals	Contains the journals in which the citation titles are published
Citation Years	The year in which the citation titles were published
Reference Titles	Contains the name of articles to which the selected paper refers to
Reference Journals	Contains the journals in which the reference titles are published
Reference Years	The year in which the reference titles were published

In the above mentioned columns, 6 columns, namely, Citation Titles, Citation Journals, Citation Years, Reference Titles, Reference Journals and Reference Years are used for collaborative filtering. All the other columns are being used for performing content-based filtering.

For scraping the data, we use selenium and scrapy packages of python. The reason for using selenium is its ability to scrape efficiently from dynamic web pages.

The data scraped from selenium is expected to fulfill all the three V's of big data,namely:

Volume of data: 3000 Records of Journal Articles are collected from four domains - Machine learning, Cloud Computing, Block Chain, Internet of Things.

Variety: Structured Data which includes:

1. Article Information
2. Popularity score
3. References and Citations Information

Velocity: According to a science blog, approximately 2.5 million new scientific papers are published each year. Articles in the dataset are published in the timespan of 2014-2019.

Various steps involved in extracting data and the output of data collection process is shown below:

1. **Importing the required python packages:**

```
# -*- coding: utf-8 -*-
from scrapy import Spider
from selenium import webdriver
from scrapy.selector import Selector
from scrapy.http import Request
from time import sleep
from selenium.common.exceptions import NoSuchElementException
```

Fig 4.1 Python packages importation

2. **Accessing google chrome through selenium webdriver:**

```
class RespapSpider(Spider):
    name = 'respap'
    allowed_domains = ['semanticscholar.org']

    def start_requests(self):
        self.driver = webdriver.Chrome(executable_path='C:/Users/abhinav/Downloads/chromedriver.exe')
```

Fig 4.2 Web driver integration with selenium

3. **Accessing the search results of Semantic Scholars through Selenium webdriver :**

```
self.driver.get('https://www.semanticscholar.org/search?year%5B0%5D=2014&year%5B1%5D=2019&publicationType%5B0%5D=JournalArticle&q=deep%20learning&sort=relevanc
```

Fig 4.3 Accessing the webpage through selenium webdriver

The above command opens the below webpage in google chrome through selenium

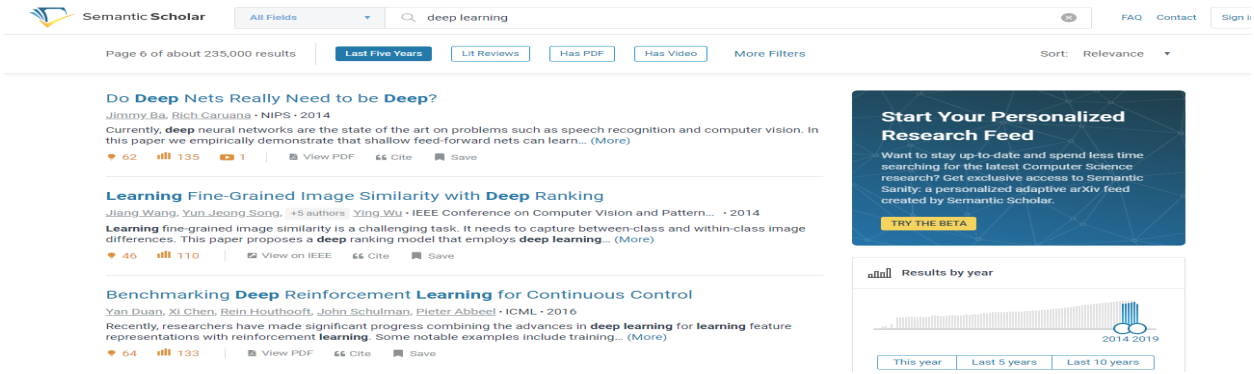


Fig4.4 The webpage accessed through selenium webdriver

4. **With the help of scrapy selector, the corresponding website link of each article in the search page is extracted:**

```
sleep(20)
sel= Selector(text=self.driver.page_source)
papers= sel.xpath('//*[@class="search-result"]/*[@class="search-result-header"]/*[@class="search-result-title"]/a/@href').extract()
for paper in papers:
    url='https://semanticscholar.org' + paper
    yield Request(url, callback=self.parse_paper)
```

Fig 4.5 Extracting the url list from the webpage accessed

Thus all the urls obtained by the end of the above code passes through parse_paper function which would extract all the necessary information from the various research paper urls

5. Extracting data from the various research paper urls:

```
def parse_paper(self, response):
    title=response.xpath('//h1/text()').extract()
    authors=response.xpath('//*[@class="subhead"]/li/*[@class="author-list"]/*[@data-heap-id=""]/*[@class="author-list__link author-list__author-name"]/*[@c
    journal=response.xpath('//*[@data-selenium-selector="venue-metadata"]/*[@class=""]/span/text()').extract()
    year_of_publication=response.xpath('//*[@data-selenium-selector="paper-year"]/*[@class=""]/span/text()').extract()
    abstract=response.xpath('//*[@class="text-truncator abstract__text text--preline"]/text()').extract()
    no_of_citations=response.xpath('//*[@class="scorecard_stat__headline__dark"]/text()').extract()
    highly_influenced_papers=response.xpath('//*[@class="scorecard__description__highlight"]/*[@class="scorecard__description__number"]/text()').extract()
    cite_background=response.xpath('//*[@data-heap-key="background"]/*[@class="scorecard__description__number"]/text()').extract()
    cite_method=response.xpath('//*[@data-heap-key="methodology"]/*[@class="scorecard__description__number"]/text()').extract()
    cite_result=response.xpath('//*[@data-heap-key="result"]/*[@class="scorecard__description__number"]/text()').extract()
    twitter_mention=response.xpath('//*[@class="scorecard_stat__headline"]/text()').extract()
    citation_titles=response.xpath('//*[@id="citing-papers"]/*[@class="card-content card-content__children-own-layout"]/*[@class="card-content-main"]/*[@cla
    citation_journals = response.xpath('//*[@id="citing-papers"]/*[@class="card-content card-content__children-own-layout"]/*[@class="card-content-main"]/*[
    citation_dates = response.xpath('//*[@id="citing-papers"]/*[@class="card-content card-content__children-own-layout"]/*[@class="card-content-main"]/*[@cl
    reference_titles= response.xpath('//*[@id="references"]/*[@class="card-content"]/*[@class="paper-detail-content-card"]/*[@class="citation-list__citation
    reference_journals= response.xpath('//*[@id="references"]/*[@class="card-content"]/*[@class="paper-detail-content-card"]/*[@class="citation-list__citati
    reference_dates= response.xpath('//*[@id="references"]/*[@class="card-content"]/*[@class="paper-detail-content-card"]/*[@class="citation-list__citations
    yield {'Title of the Article':title,'Authors':authors,'Published Journal':journal,"Year of Publication":year_of_publication,"Abstract of the article":ab
```

Fig 4.6 Extracting the data list from the URLs extracted

This function would yield the data extracted for various variables in the form of dictionaries

6. Execution of the above set of python script and saving the output in csv file:

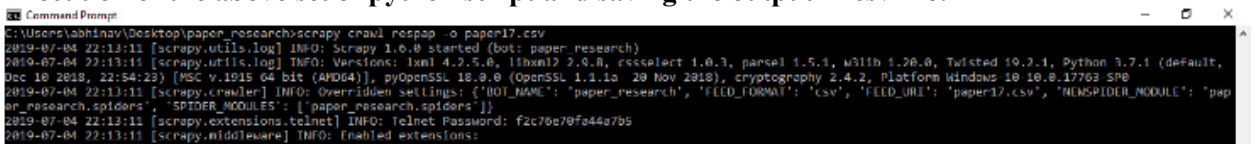


Fig 4.7 Executing the above code through scrapy crawler

The data scraped in the form of dictionaries displayed in command prompt

Fig 4.8 Output of the scrapy crawler in the form of dictionary

7. Final CSV file obtained through web scraping as displayed by pandas dataframe:

	Title of the Article	Authors	Published Journal	Year of Publication	Abstract of the article	Number of Citations	Highly Influenced Papers	Cite Background	Cite Methods	Cite Results	Twitter Mentions
0	Practical Black-Box Attacks against Machine Le...	Nicolas Papernot,Patrick D. McDaniel,Ananthram...	AsiaCCS	2017	Machine learning (ML) models, e.g., deep neura...	472 Citations	68	346	131	8.0	NaN
1	Scaling Distributed Machine Learning with the ...	Mu Li,David G. Andersen,Bor-Yiing Su	OSDI	2014	Big data may contain big values, but also brin...	472 Citations	109	249	250	13.0	NaN
2	Fashion-MNIST: a Novel Image Dataset for Bench...	Hua Xiao,Kashif Rasul,Roland Vollgraf	ArXiv	2017	We present Fashion-MNIST, a new dataset compri...	488 Citations	166	85	342	11.0	87.0

Fig 4.9 Content-based dataframe output

8. The citations and reference information in the final pandas dataframe:

	Title of the Article	Authors	Year of Publication	Citations Titles	Citations Journals	Citations Dates
0	TensorFlow: Large-Scale Machine Learning on He...	Martín Abadi,Ashish Agarwal,Xiaoqiang Zheng	2015	A Machine Learning Based Morphological Classif...	BMC Bioinformatics,IMWUT,2019 19th IEEE/ACM In...	2019,2019,2019,2019,2019,2019,2019,2019,2019,2019
1	Scaling Distributed Machine Learning with the ...	Mu Li,David G. Andersen,Bor-Yiing Su	2014	SGD: Decentralized Byzantine Resilience,Effici...	ArXiv,PVLDB,EuroSys,NIPS,Cluster Computing,ArXiv	2019,2018,2017,2017,2016,2016,2014,2019,2019,2019
2	Fashion-MNIST: a Novel Image Dataset for Bench...	Han Xiao,Kashif Rasul,Roland Vollgraf	2017	2018-1203 Parameter Continuation with Secant A...	FAT,ArXiv,ICML,2019 Design, Automation & Test ...	2019,2019,2019,2019,2019,2019,2019,2019,2019,2019
3	Asynchronous Methods for Deep Reinforcement Le...	Volodymyr Mnih,Adrià Puigdomènech Badia,Koray ...	2016	ACTRCE: Augmenting Experience via Teacher's Ad...	ArXiv,ArXiv,ArXiv,ACL 2019,IEEE transactions o...	2019,2019,2019,2019,2019,2019,2019,2019,2019,2019
4	Practical Black-Box Attacks against Machine Le...	Nicolas Papernot,Patrick D. McDaniel,Ananthram...	2016	Adversarial Attacks on Remote User Authenticat...	ArXiv,ArXiv,ICML,ArXiv,IEEE Transactions on Ne...	2019,2019,2019,2019,2019,2019,2019,2019,2019,2018
5	MLlib: Machine Learning in Apache Spark	Xiangrui Meng,Joseph K. Bradley,Ameet Talwalkar	2016	A Workload-aware Resource Management and Sched...	2018 International Conference on Intelligent A...	2019,2018,2018,2017,2017,2017,2016,2016,2016,2015

Fig 4.10 Collaborative-based dataframe output for citations

Reference Titles	Reference Journals	Reference Dates
Large Scale Distributed Deep Networks,Batch No...	NIPS,ICML,Neural Networks,EuroSys,ArXiv,ArXiv,...	2012,2015,2015,2015,2015,2015,2015,2015,2015,2015
An Architecture for Parallel Topic Models,Scal...	PVLDB,WSDM,Technical Talk,	2010,2012,2012
Multi-column deep neural networks for image cl...	2012 IEEE Conference on Computer Vision and Pa...	2012,2017,2013,2009,2009,1998
Prioritized Experience Replay,Deep Reinforceme...	ICLR,AAAI,IEEE Transactions on Neural Networks...	2016,2015,1988,2016,2011,1992,2016,2015,2015,2015
Man vs. computer: Benchmarking machine learnin...	Neural Networks,In Proceedings of the 1st IEEE...	2012,2016,2015,2014
TuPAQ: An Efficient Planner for Large-scale Pr...	https://www.edx. org/course/scalable-machine-l...	2015,2015,2014,2013,2013,2013,2012,2009,2009

Fig 4.11 Collaborative-based dataframe output for references

This information has been extracted from the following section of semantic scholar

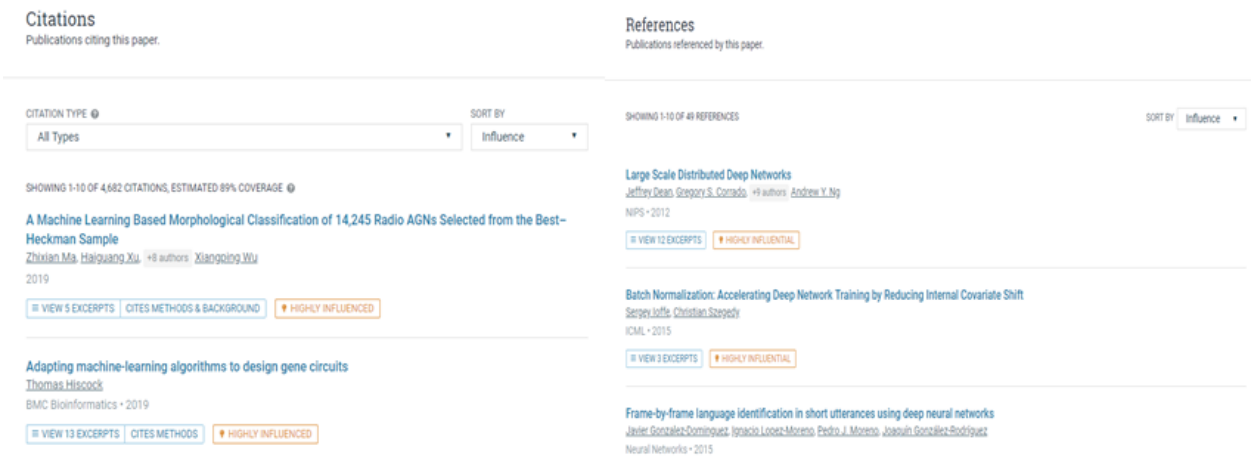


Fig 4.12 Webpage sections from which the data for collaborative filtering is extracted

5. DATA PRE-PROCESSING

Data preprocessing is the data mining technique to transform the raw real-world data into a meaningful format. Usually the data collected from real world will be noisy, incomplete and inconsistent. In the data pre-processing step, data will be cleaned and modified according to the project needs.

In our project, we analyzed the data collected from Semantic Scholar website. The data contained many missing values. Missing data in each column of .csv file was handled differently as explained below.

- There were missing data in the authors and published journal columns. As the research paper without this information was not reliable data, we removed these rows of data from our dataset.
- There were missing data in citations, highly influenced papers, cite Backgrounds, Twitter Mentions columns also. These columns were filled with zero , because missing data was interpreted as zero citations, zero twitter mentions etc.

6. DATA VISUALIZATION

Data Visualization is very helpful to communicate information about the data very clearly and efficiently. We plotted some of the below graphs to visualize the data.

6.1 Network graph plot for dataset:

To analyse the data, a network graph is plotted for the dataset using Gephi. In figure Fig(6.1.1) the nodes are in black and the edges are in red color. Figure Fig(6.1.2) is network graph plotted for a subset of data from the dataset to show clear connections between the data nodes and their edges. From Fig(6.1.2) we can observe how domains research papers in dataset are related to the year they were published in and also how some domains are closely connected compared to others.

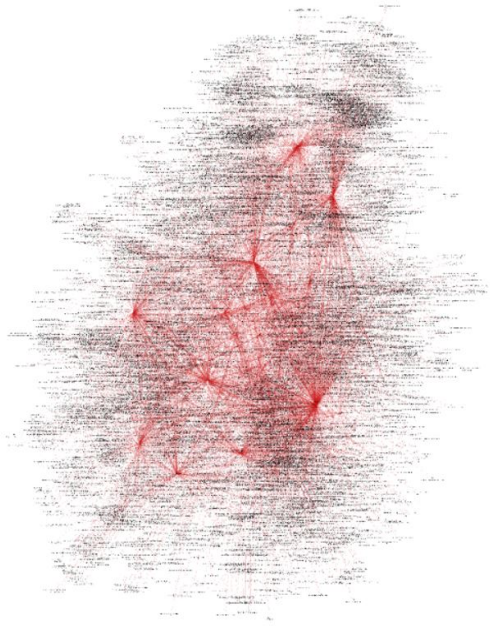


Fig 6.1.1 Network graph of dataset

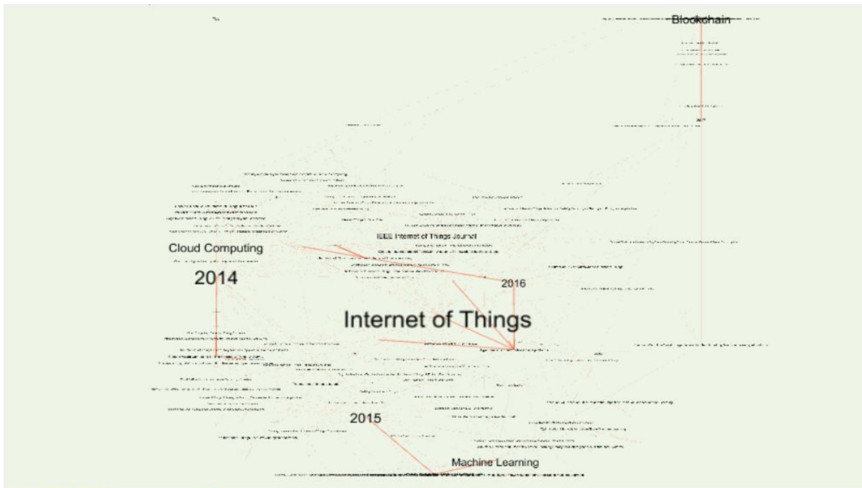


Fig 6.1.2 Network graph for a subset of data

6.2. **Data visualization using word cloud:**

In order to get the bird eyes' overview, we used Word Cloud over the title.

Fig 6.2 World Cloud

6.3. **Number of research papers versus available Domain of paper:**

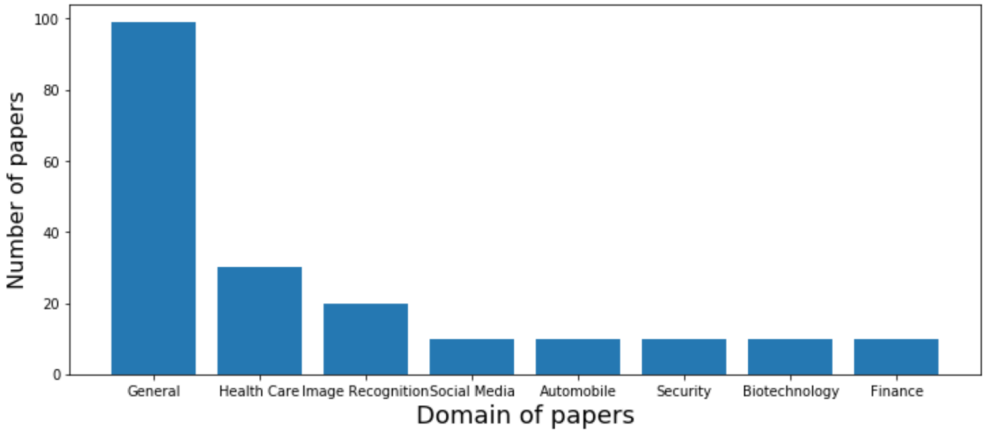


Fig 6.3 Bar graph representing domain vs research papers over a subset of data

6.4. Number of research papers versus Technology of available papers.

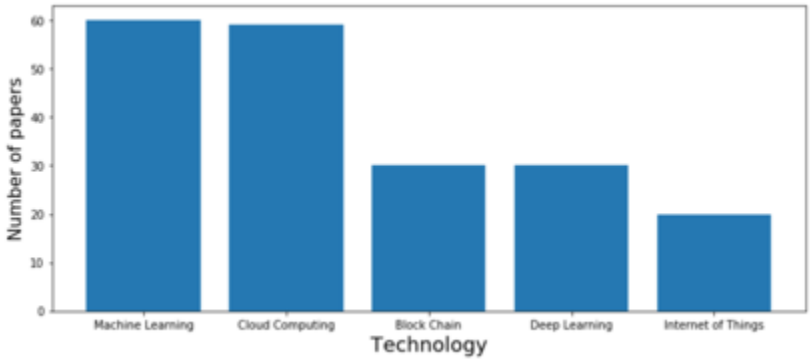


Fig 6.4 Bar graph representing technology vs research papers over a subset of data

6.5. Number of citations Versus Technology of available papers.

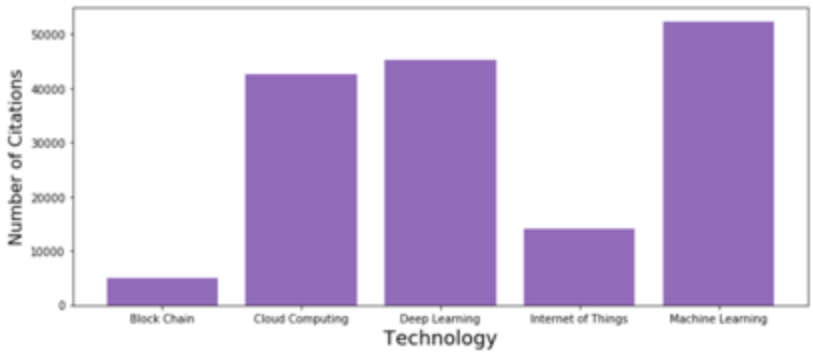


Fig 6.5 Bar graph representing citation vs technology over a subset of data

7. TECHNICAL IMPLEMENTATION

In this section, we will see the implementation of the models discussed earlier to build our Recommendation System. Our Hybrid Recommender System is built as a cascade model combining the strengths of Content-based approach and Collaborative Filtering approach in a pipeline to give the best recommendations to the user.

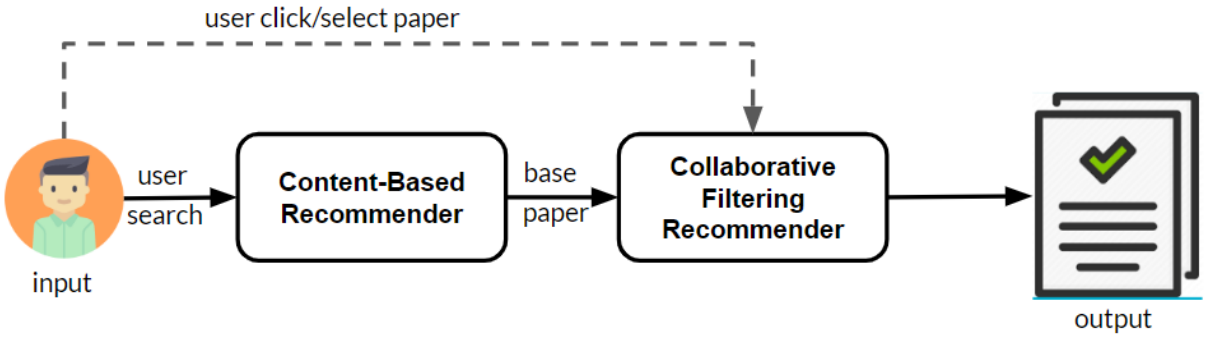


Fig 7.1 Pipeline Hybrid model

(i) Content-based recommender model:

In this model we have used the TF-IDF (term frequency-inverse document frequency) for information retrieval and text mining. The text from the user search is fetched and stop words are removed to find keywords from it. We then proceed to find the frequency score for each of the words from keywords in the available corpus. Title and abstract of the articles are used to calculate the TF-IDF scores as our dataset doesn't contain the whole document text. To provide more importance to the title over abstract we have calculated weighted sum of the TF-IDF score, giving title 75% weightage and abstract 25% weightage.

$$TF\text{-}IDF(i,j) = TF(i,j) * IDF(i)$$

where, i is a keyword and j is the document.

TF(i,j): the term frequency of a keyword i in a document j

IDF(i): the inverse document frequency is calculated as

$$IDF(i) = \log N/(n(i))$$

where, N is the number of recommendable documents and n(i) is the number of documents from N in which the keyword i appears

We then arrive with recommendations from the content similarity from user search to the available papers in corpus. However, to enhance the recommendations we ranked them using some popularity measures available in the corpus. Citation Score, Influence factor(highly influenced papers score), Twitter mentions data available in the corpus is used with a weightage to provide score to the recommendations in order to rank them. The weightages provided to citation score, influence factor and twitter mentions are 40%, 40% and 30% respectively. The weightage is decided based on multiple iterations to obtain best recommendations.

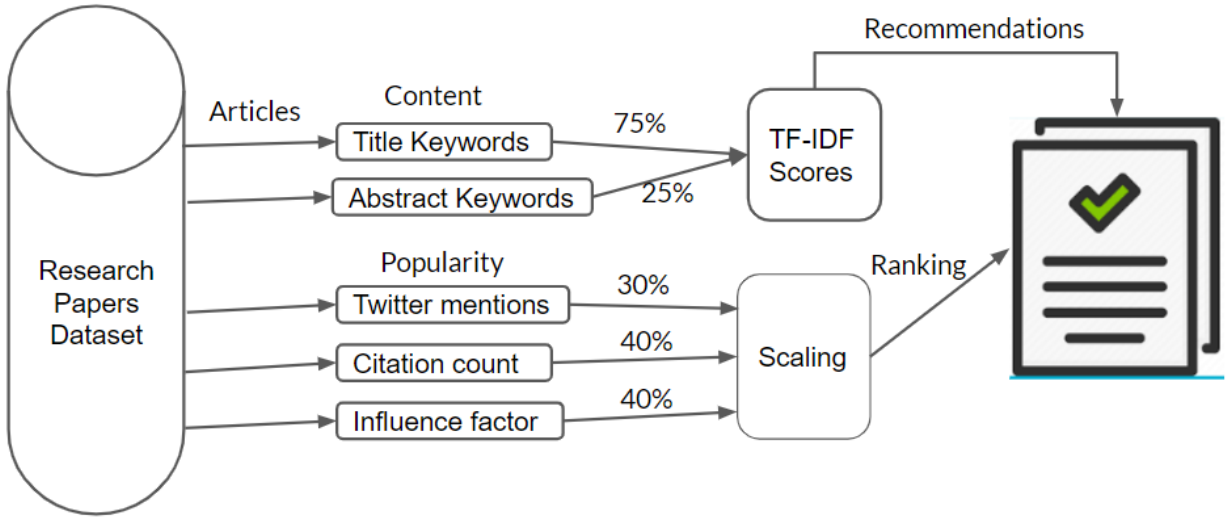


Fig 7.2 Content-based recommender model

NLTK library is used to tokenize and remove stop words from the user search. TfidfVectorizer from scikit learn libraries is used to convert the keywords in corpus to a matrix of TF-IDF features. The keywords from the search are then matched with this matrix to identify their TF-IDF scores. The title keywords TF-IDF scores are multiplied by the fraction 0.75 and abstract keywords TF-IDF scores are multiplied by the fraction 0.25 and then their sum is calculated to be the final TF-IDF scores. The top ten documents with the highest TF-IDF scores are considered the recommendations for the search. Further the popularity scores for these top ten recommendations are calculated to rank them accordingly. To calculate the citation score, the citation score of the document is divided by the maximum value available for citation score in the corpus and then multiplied by 100 and the factor of weightage(which is 0.4 as citation score constitutes 40% of the total score). The influence factor score and twitter mentions score to these documents are calculated similarly. The result recommendations would thus have the most popular research paper on the top of the list, along with being one of the best matches for the user search keywords.

Let us consider an example search to see how our model works. Assuming the user searches with the keyword “tensorflow”, keywords extracted from search would be “tensorflow” after removing stop words. Fig(7.3) shows the TF-IDF matrix to which the word is matched.

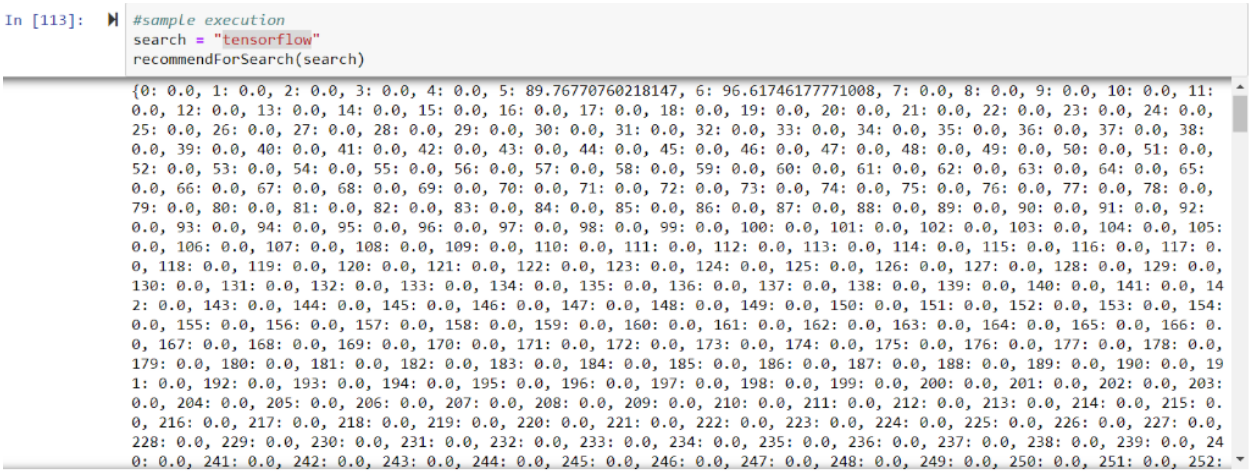


Fig 7.3 TF-IDF scores of sample search

By observing the matrix we can see that the documents having the keyword match are the ones at index 5, 6 and so on. Fig(7.4) shows the top ten matches based on highest TF-IDF scores and their popularity scores.

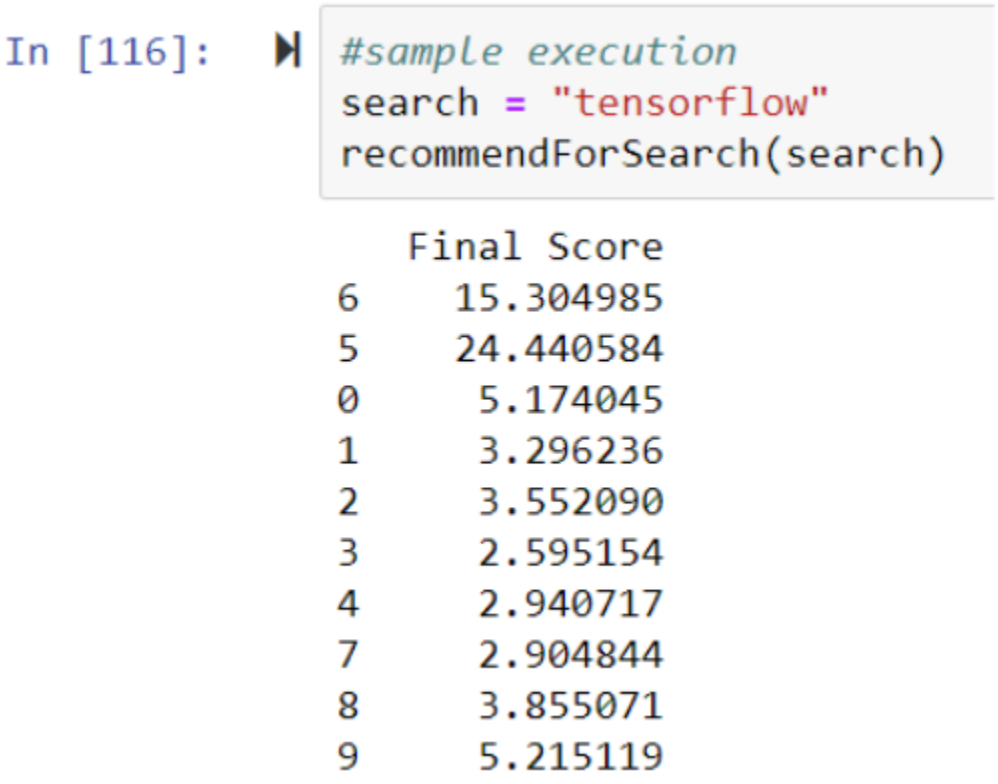


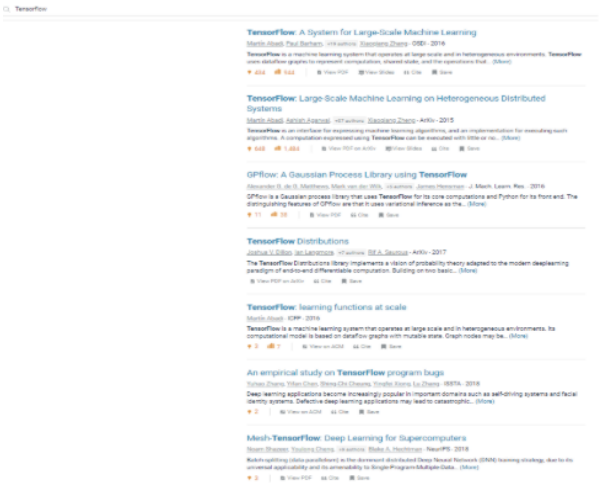
Fig 7.4 Popularity scores measure of sample search

```
In [101]:  #sample execution
           search = "tensorflow"
           print(recommendForSearch(search))
```

```
5    TensorFlow: Large-Scale Machine Learning on He...
6    TensorFlow: A System for Large-Scale Machine L...
9    Convolutional LSTM Network: A Machine Learning...
0    Fashion-MNIST: a Novel Image Dataset for Bench...
8    MXNet: A Flexible and Efficient Machine Learni...
2    Scaling Distributed Machine Learning with the ...
1            MLlib: Machine Learning in Apache Spark
4         DaDianNao: A Machine-Learning Supercomputer
7    DianNao: a small-footprint high-throughput acc...
3    Practical Black-Box Attacks against Machine Le...
Name: Title of the Article, dtype: object
```

Fig 7.5 Content-based model recommendations for sample search

From the figures Fig(7.4) and Fig(7.5) we can observe that the final results in Fig(7.5) are ranked based on the popularity scores from Fig(7.4). Also, we can observe the user search keyword match in the top two titles in recommendations.

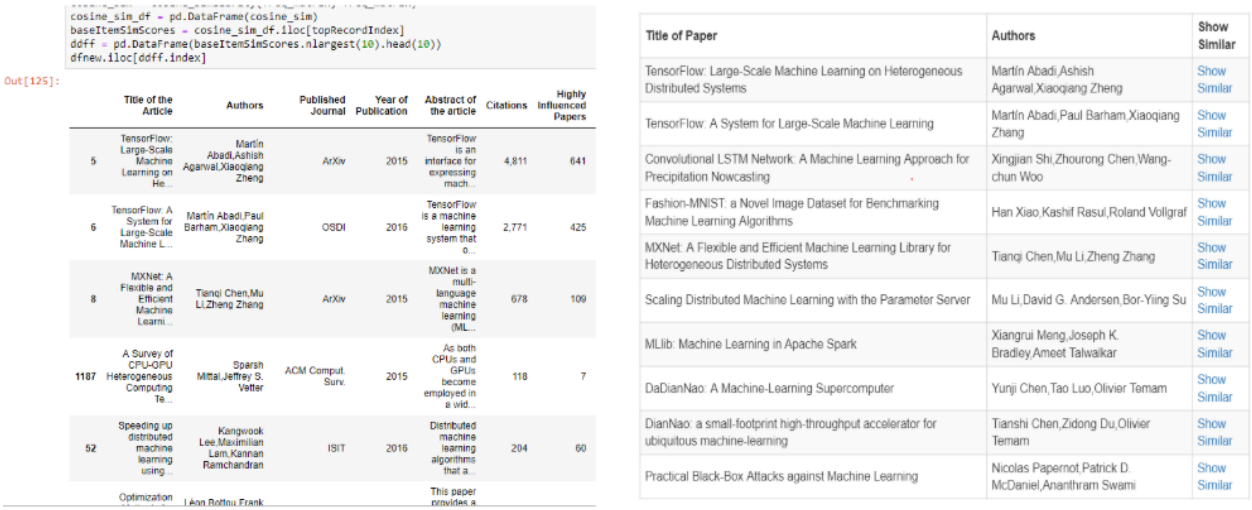


Title of Paper	Authors	Show Similar
TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems	Martín Abadi,Ashish Agarwal,Xiaoqiang Zheng	Show Similar
TensorFlow: A System for Large-Scale Machine Learning	Martín Abadi,Paul Barham,Xiaoqiang Zhang	Show Similar
Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting	Xingjian Shi,Zhourong Chen,Wang-chun Woo	Show Similar
Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms	Han Xiao,Kashif Rasul,Roland Vollgraf	Show Similar
MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems	Tianqi Chen,Mu Li,Zheng Zhang	Show Similar
Scaling Distributed Machine Learning with the Parameter Server	Mu Li,David G. Andersen,Bor-Yiing Su	Show Similar
MLlib: Machine Learning in Apache Spark	Xiangrui Meng,Joseph K. Bradley,Ameet Talwalkar	Show Similar
DaDianNao: A Machine-Learning Supercomputer	Yunji Chen,Tao Luo,Olivier Temam	Show Similar
DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning	Tianshi Chen,Zidong Du,Olivier Temam	Show Similar
Practical Black-Box Attacks against Machine Learning	Nicolas Papernot,Patrick D. McDaniel,Ananthram Swami	Show Similar

Fig 7.6 Comparison of results with semantic scholar

A comparison is made between the model results and the semantic scholar website from which the corpus is obtained to verify the efficiency of recommendation results. In figure Fig(4.6) we can see on the left the search results of the keyword “tensorflow” in semantic scholar website and on the right the search results of the same keyword in our model. The search results from the semantic scholar highlights the keyword “tensorflow” in all the titles in results. However, in our model we have relatively less data in the corpus and also the keyword match is done in abstract along with the title. Hence, the recommendations differ. If

we observe the first two titles in both the results are same but ranked differently. The semantic scholar result record1 has comparatively less popularity score than record2 (popularity scores of the articles can be seen below the abstract in numbers). Our model ranks the popular one to be on the top.



```
cosine_sim_df = pd.DataFrame(cosine_sim)
baseItemSimScores = cosine_sim_df.iloc[topRecordIndex]
ddff = pd.DataFrame(baseItemSimScores.nlargest(10).head(10))
dfnew.iloc[ddff.index]
```

Out[125]:

	Title of the Article	Authors	Published Journal	Year of Publication	Abstract of the article	Citations	Highly Influenced Papers
5	TensorFlow: Large-Scale Machine Learning on He...	Martín Abadi,Ashish Agarwal,Xiaoqiang Zheng	ArXiv	2015	TensorFlow is an interface for expressing mach...	4,811	641
6	TensorFlow: A System for Large-Scale Machine L...	Martín Abadi,Paul Barham,Xiaoqiang Zhang	OSDI	2016	TensorFlow is a machine learning system that o...	2,771	425
8	MXNet: A Flexible and Efficient Machine Learni...	Tianqi Chen,Mu Li,Zheng Zhang	ArXiv	2015	MXNet is a multi-language machine learning (ML...	678	109
1187	A Survey of CPU-GPU Heterogeneous Computing Te...	Sparsh Mittal,Jeffrey S. Vetter	ACM Comput. Surv.	2015	As both CPUs and GPUs become employed in a wid...	118	7
52	Speeding up distributed machine learning using...	Kangwook Lee,Maximilian Lam,Kannan Ramchandran	ISIT	2016	Distributed machine learning algorithms that a...	204	60
	Optimization	Léon Bottou,Frank			This paper provides a		

Title of Paper	Authors	Show Similar
TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems	Martín Abadi,Ashish Agarwal,Xiaoqiang Zheng	Show Similar
TensorFlow: A System for Large-Scale Machine Learning	Martín Abadi,Paul Barham,Xiaoqiang Zhang	Show Similar
Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting	Xingjian Shi,Zhourong Chen,Wang-chun Woo	Show Similar
Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms	Han Xiao,Kashif Rasul,Roland Vollgraf	Show Similar
MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems	Tianqi Chen,Mu Li,Zheng Zhang	Show Similar
Scaling Distributed Machine Learning with the Parameter Server	Mu Li,David G. Andersen,Bor-Yiing Su	Show Similar
MLlib: Machine Learning in Apache Spark	Xiangrui Meng,Joseph K. Bradley,Ameet Talwalkar	Show Similar
DaDianNao: A Machine-Learning Supercomputer	Yunji Chen,Tao Luo,Olivier Temam	Show Similar
DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning	Tianshi Chen,Zidong Du,Olivier Temam	Show Similar
Practical Black-Box Attacks against Machine Learning	Nicolas Papernot,Patrick D. McDaniel,Ananthram Swami	Show Similar

Fig 7.7 Comparison of results with cosine similarity

Another comparison is made by finding recommendations using cosine similarity between the best TF-IDF match with all the documents in the corpus. The recommendation results from cosine similarity are seen on the left side in figure Fig(7.7). We can observe that the record with index 52 has more popularity than the record with 1187, but it is ranked below than the latter. Also, it is not very accurate to calculate the cosine similarity between the first document with the rest as it may have matched with many other words in it, while user search was specifically for “tensorflow”.

(ii) Collaborative Filtering recommender model:

In this model, item-based filtering is performed to obtain recommendations. The model doesn’t use any user data for performing item-based filtering, but uses citation analysis to find the similarity between documents. The base paper for which we look for similar papers is obtained through user’s click(implicit). Based on the previous model we first provide user with recommendations based on content from his search query and we also provide an option for the user to select more of which among them he would like to see. The user could click “see similar” on the paper which he feels is most relevant to his search and we consider that paper to be our base paper to do a collaborative item based filtering on the corpus. The algorithm discussed in design section is simplified as below for implementation of collaborative item based filtering.

1.(base paper) 2.(Potential recommendable paper) 3. (Another paper)

If ((2 has cited papers cited in 1) && (3 has cited both 1 and 2))
then (use 2 to recommend for those who view 1)

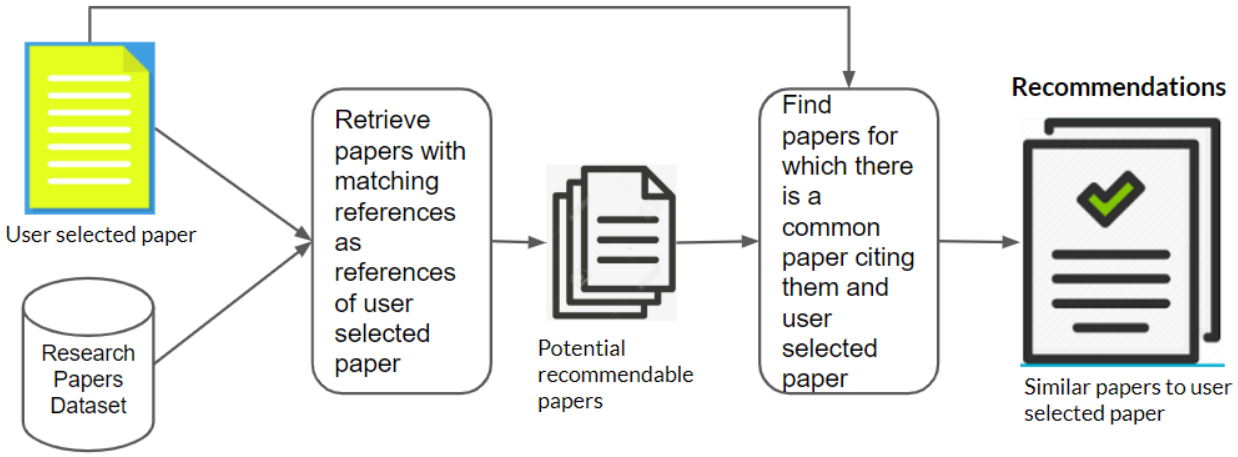


Fig 7.8 Collaborative Filtering recommender model

We first check all the potential recommendable papers from the corpus based on the first condition here in the if check. To do this we match every document's references in the corpus with the references of the base paper to identify documents which have at least one reference in common. Once we have the list of potential recommendable papers, we further check the second condition in the if check to obtain the final list of papers that can be recommended. Through this citation match analysis we try to find the best close documents which could be used together or are related to one another. The more papers that have cited them together the better close they are.

```
In [101]:  #sample execution
           search = "tensorflow"
           print(recommendForSearch(search))

           5    TensorFlow: Large-Scale Machine Learning on He...
findMoreSimilar(5, filename)

Out[10]: ['TensorFlow: A System for Large-Scale Machine Learning',
          'MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems',
          'Deep Learning',
          'Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks',
          'Convolutional Networks on Graphs for Learning Molecular Fingerprints',
          'Learning Fine-Grained Image Similarity with Deep Ranking',
          'Deep Reinforcement Learning with Double Q-learning',
          'Deep Residual Learning for Image Recognition',
          'Big Data Deep Learning: Challenges and Perspectives',
          'Federated Optimization: Distributed Machine Learning for On-Device Intelligence',
          'Project Adam: Building an Efficient and Scalable Deep Learning Training System',
          'Massively Parallel Methods for Deep Reinforcement Learning']
```

Fig 7.9 Collaborative Filtering recommendations

A sample execution of this model is shown in figure Fig(7.9). From the previous content-based recommendations, let us assume user wants to see more similar papers as of the first recommended research paper. The output for similar papers for this paper is shown in figure Fig(7.9). The advantage of this model is user need not fine tune his search query, but would rather see options of available relevant documents to his search. By cascading this model to the previous one a refined set of recommendations which are more relevant to user's search are provided.

8. CLOUD DEPLOYMENT on GCP

In our project, after having a working prototype, we built a complete end to end working app (or a website) that we can deploy on cloud, so that we can provide the paper recommendation system as a service.

In this project, our backend was developed using Python and so we made use of Flask, which is a lightweight web framework. It is a third-party Python library used for developing web applications.
This framework uses Jinja template engine and so it was well suited for our needs. To have an interactive and responsive User Interface, we made use of Bootstrap. Bootstrap is industry standard for making web responsive apps.

Now, coming to the cloud deployment, for this project we made an analysis of Amazon Web Service, popularly called as AWS and Google Cloud Platform, popularly called as GCP.

GCP is very user friendly and easy to use compared to AWS. Also, Google provides $300 credit for students to make use of their platform. In Google starting of the App Engine, building the app and deploying the app is user friendly and less of boiler plate process.

Once the website was deployed on the GCP, it is made accessible on https://paper-recommendation-system.appspot.com and we are seeing that the requests are getting served very well.

8.1 Screenshots of the Website hosted on GCP

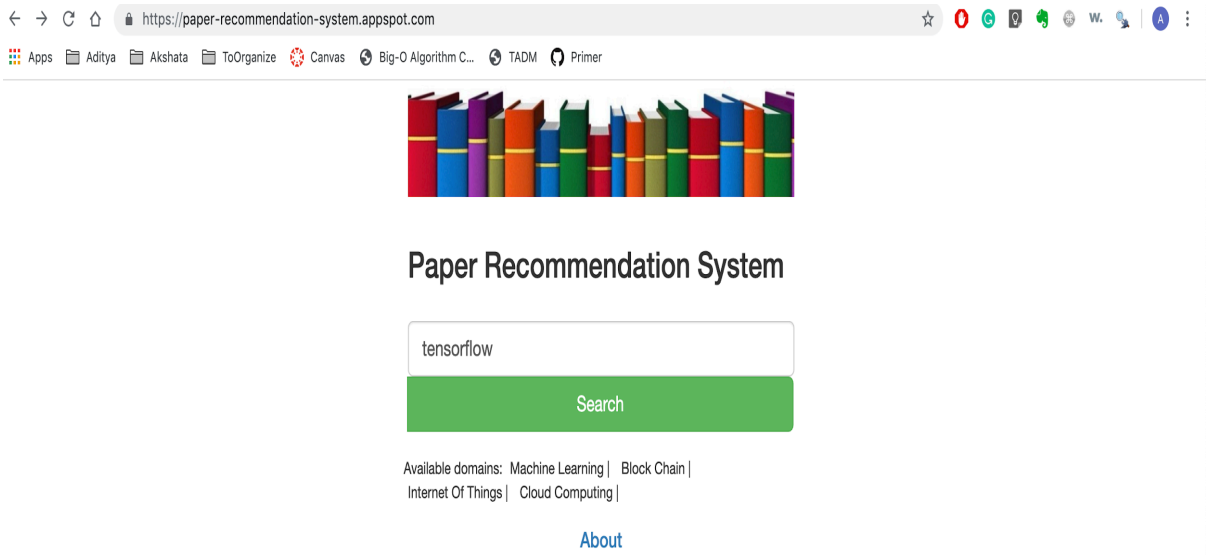


Fig 8.1 Landing web page of the Paper Recommendation System on GCP

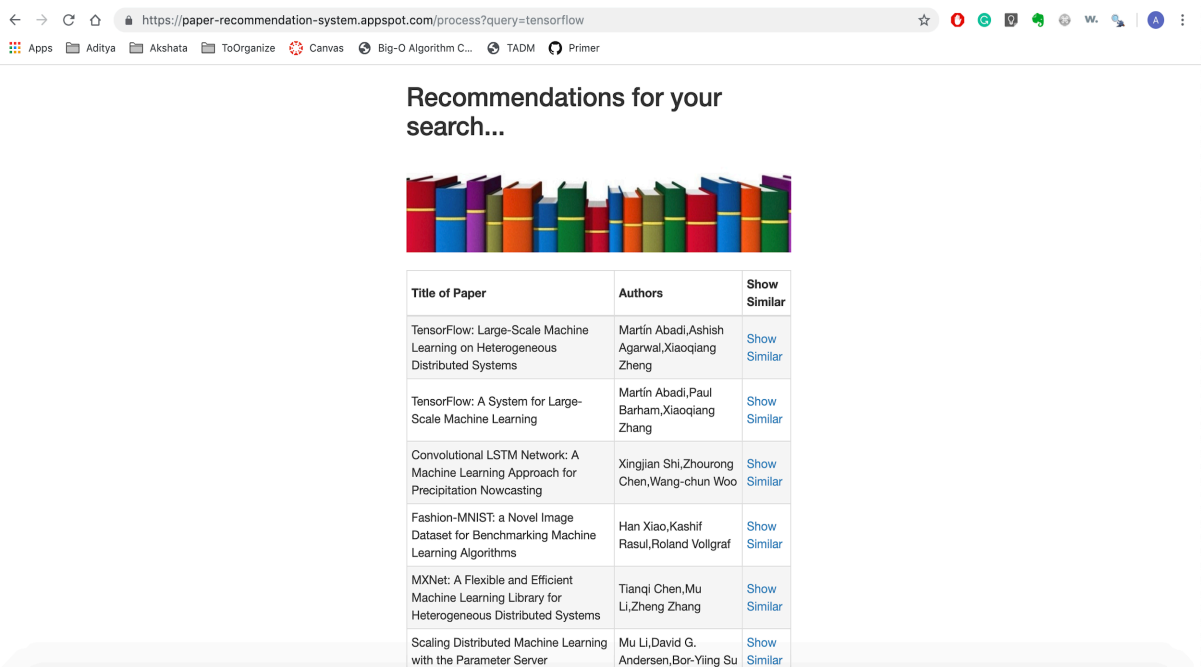


Fig8.2 Results from the Recommendation System

9. EVALUATION:

The Evaluation metric is an essential part for evaluating the results of any recommendation system, because it provides a direct insight into the comparison of competing recommendation systems and ranks them accordingly. The two evaluations that we have used are Discounted cumulative Gain and Peer Reviews.

Discounted cumulative gain(DCG) Evaluation:

Several evaluation metrics are widely used in comparing different recommendation systems such as precision and recall, mean average precision (MAP) and discounted cumulative gain (DCG). Among those metrics, DCG has become most popular for comparing the performance of ranking functions. In this article, we focus on the critical issue of determining the gain and discount factors used in the definition of DCG. The DCG evaluated for its top K documents is defined as

$$DCG_{g,K}(\pi) = \sum_{i=1}^{K} c_i g_{l_{\pi(i)}}, \quad K = 1, \ldots, N,$$

where glπ(i) indicates the gain value for the rank i document xπ(i), The set of parameters c1 > c2 > ⋯ > ck > 0 are the so-called discount factors. This reflects the fact that users emphasize more the documents on the top of the rankings

$$g_i = 2^i - 1 \qquad c_k = \frac{1}{\log(k+1)}$$

In DCG calculation, we first calculated the similarity scores of all the documents with respect to the user search keyword. For example: we have taken “machine learning” as user search data and found out the

top 10 recommendations for it using our recommendation system and also from the semantic scholar website. we also have calculated the similarity scores of these documents based on our similarity score calculation metric and got the following results:

Our recommendation system Top 10 recommendations:

Title of the Paper	Sim_Scores
Machine Learning in Medicine.	24.4405
Machine Learning for Medical Imaging.	22.3107
Quantum-enhanced machine learning	21.7457
Diversity in Machine Learning	20.9807
Machine learning for engineering	20.4365
Trustless Machine Learning Contracts; Evaluating and Exchanging Machine Learning Models on the Ethereum Blockchain	19.6865
Machine Learning Meets Databases	17.3401
Transformative Machine Learning	16.5403
Extreme learning machine based supervised subspace learning	16.1012
Two-stage Optimization for Machine Learning Workflow	15.9038

Semantic Scholar top 10 recommendations:

Gaussian processes for machine learning	14.0165
Learning Deep Architectures for AI	13.9356
An Introduction to MCMC for Machine Learning	13.5456
Machine Learning for the Detection of Oil Spills in Satellite Radar Images	13.4125
Machine Learning - a probabilistic perspective	12.7583
Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach	12.3635
Data mining - practical machine learning tools and techniques, Second Edition	12.0618
Bioinformatics - the machine learning approach	11.3709
Introduction to machine learning	11.1723

Bayesian reasoning and machine learning	9.0756

Here, we assumed the following:

1) similarity scores > 20 - “perfect” match for the user search – value = 4
2) 10 < similarity scores < 20 – “good match” – value = 3
3) similarity scores < 10 – “average match” – value = 2

Also, each document is assigned a gain value(g_i) and a discount value(Ck) decreasing from top to bottom of the recommendation list defined by the formula:

$$g_i = 2^i - 1 \qquad c_k = \frac{1}{\log(k+1)}$$

Where “i” is the value given to each class of the document as assumed above.
Where “k” is given by the rank of the document in the list.

The following are the results of the DCG evaluation of our recommendation system and semantic scholar website.

```
In [3]: import math
        import numpy as np

        def get_dcg(k,l):
            c = [1/math.log(i+2,10) for i in range(k)]
            print('Discount: ', c)
            g = [np.power(2,j)-1 for j in l]
            print('Gain: ',g)
            DCG = [p*q for p,q in zip(c,g)]
            DCG = np.sum(DCG)
            return DCG

        print('DCG of our Recommendation System: ', get_dcg(10, [4,4,4,4,4,3,3,3,3,3]))

        Discount:  [3.321928094887363, 2.095903274289385, 1.6609640474436815, 1.4306765580733931, 1.285097208938469, 1.183294
        6624549385, 1.1073093649624544, 1.0479516371446924, 1.0, 0.9602525677891276]
        Gain:  [15, 15, 15, 15, 15, 7, 7, 7, 7, 7]
        DCG of our Recommendation System:  184.01019538094286

In [4]: print('DCG of semantic scholar: ', get_dcg(10, [3,3,3,3,3,3,3,3,3,2]))

        Discount:  [3.321928094887363, 2.095903274289385, 1.6609640474436815, 1.4306765580733931, 1.285097208938469, 1.183294
        6624549385, 1.1073093649624544, 1.0479516371446924, 1.0, 0.9602525677891276]
        Gain:  [7, 7, 7, 7, 7, 7, 7, 7, 7, 3]
        DCG of semantic scholar:  101.81263164072803
```

Fig 9.1

Peer review:
The second evaluation that we have used is peer review evaluation, in which we created a google form with some feedback questions and record the responses in a pie chart. The following pie charts are obtained as reviewed by the peers:

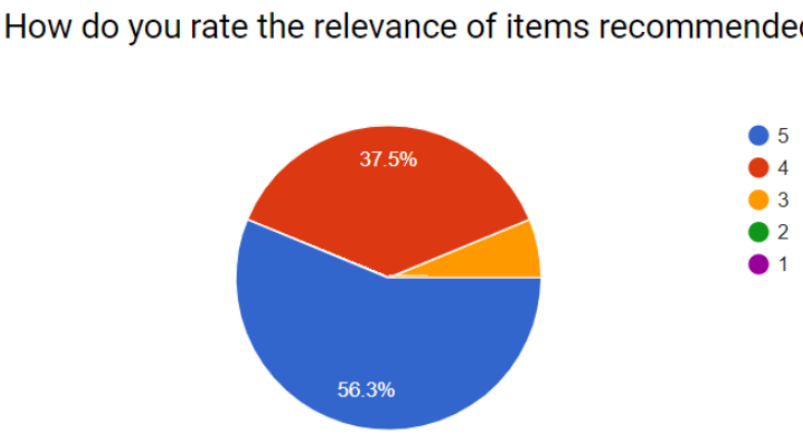


Fig 9.2

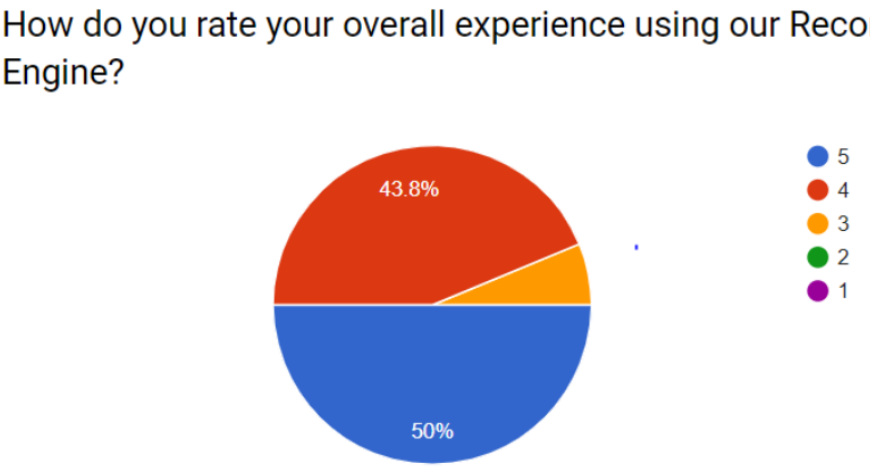


Fig 9.3

10. CONCLUSION AND FUTURE WORK

In this project we developed hybrid pipelined recommendation system to recommend most similar research papers. Hybrid recommender system consists of content based and collaborative methods of recommender system. Content based recommender system identifies similarity of papers based on content of paper using information retrieval and text mining. In collaborative method, we utilized the publicly available contextual metadata to leverage the advantages of collaborative filtering approach in recommending a set of related papers to a researcher based on paper-citation relations. The approach mined the hidden associations between a research paper and its references and citations using paper-citation relations.

We can improve the computation time by building the model offline and then use this model for recommendation system. Also, we believe that this model can be built using Deep Neural Networks like Recurrent Neural Networks.

11. INDIVIDUAL CONTRIBUTIONS

Each team member contributed towards literature survey, recommendation system design as a group along with report writing and PowerPoint presentations.

A. Abhinav Balasubramanian: Performed Data Scraping from Semantic Scholar and involved in the implementation of the content-based and collaborative recommendation model.

B. Naga Janaki Dwadasi: Plotted Network Graphs for analysis of dataset. Implemented the hybrid recommendation system design which includes content-based model and the collaborative model in python.

C. Akshata Kulkarni: Implemented Data Pre-Processing, Data visualization. Implemented the Web app that can be deployed on Google Cloud Platform using Python, Flask and Bootstrap.

D. Manish Katturu: Designed and helped in implementing the UI and performed the evaluations, involved in the implementation of the content-based.

12. REFERENCES

1. Haruna K, Akmar Ismail M, Damiasih D, Sutopo J, Herawan T (2017) A collaborative approach for research paper recommender system. PLoS ONE 12(10): e0184516.

2. Murali, M Viswa, Vishnu, T G, and Victor, Nancy. 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS). 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS). IEEE, 2019.

3. https://gephi.org/