Hybrid System for Research Paper Recommendation and Text Summarization

Chitransh Bose
chitransh22096@iiitd.ac.in
IIITD

Ranit Pal
ranit22096@iiitd.ac.in
IIITD

Soumyajyoti Das
soumyajyoti22075@iiitd.ac.in
IIITD

Varun Jhunjhunwala
varun22087@iiitd.ac.in
IIITD

1. Problem statement

We are working with a "Hybrid system for Research paper recommendation and Text summarization" that will be basically recommend the user with the top 'k' research paper based on the user input along with the short summary and based on the summary obtained it will re rank the research papers. Thus our work will be divided into four parts first will be recommendation of relevant research papers and the other will be summarization of text and then the summarised text will be used for the re-ranking of the papers and finally the abstractive summary will be provided. We have used the content- based filtering in our approach to recommend the papers and the combination of extractive and abstractive summarization techniques for the summarization part.

2. Motivation

As the use of internet id growing exponentially the amount of online data is also being generated at an exponential rate. Thus exploring the information that is relevant to user is a time consuming and a tedious task. Since, the research community is working dedicatedly, the websites that provide research resources are of utmost use and retrieving all relevant information will thus consume a lot of time. Therefore, we need a system that can resolve this issue.

3. Literature Review

Joeran Beel et al.[1] in their paper have provided a survey based on a research paper recommendation system, in which they have identified the methods which are mostly used for recommendations along with the various shortcomings that are present in various papers like the choice of evaluation metrics, etc. Joonseok Lee et al.[2], have proposed a Personalized Academic Research Paper recommendation system that recommends the use of articles and papers related to the interest of the user by finding the similarity between the text using collaborative filtering. As there is a lot of information present over the internet these days, the extraction of relevant documents and further the relevant information is very important. Thus, S.A. Babar et al. in their paper have presented a system for text summarization. They have discussed various text summarization methods like TF-IDF, Graph-theoretic approach, Machine Learning approach, and Automatic Text Summarization based on Fuzzy logic. Julian Kupiec et al.[3] proposes a trainable document summarizer that uses a neural network architecture to generate summaries of input documents. They evaluated their approach on a benchmark dataset and compared it to existing summarization systems. S.M. Kamruzzaman et al.[4] in their work have proposed a new algorithm for the classification of text that is based on the word relation rather than the word only using a lesser number of documents for training. Further, it uses Naïve Bayes for extracting the features. Arun Kumar Yadav et.al[5] has used reinforcement learning as their base model and on the basis of that they have proposed their algorithm which they have evaluated using Bilingual Evaluation Understudy (BLEU) and Recall-Oriented Understudy for Gisting Evaluation (ROUGE).

4. Novelty

Although there are search engines and other systems available that recommends the research papers based on the user query but none of the systems are there that recommends as well as provide the summary of the paper. Reading the complete paper and then deciding if it relevant to the user or not is a time consuming task. Although abstracts are present but the abstract are less informative as they are limited to 150-200 words and lack the implementation details of the paper. Thus, we have implemented a paper recommendation and summarization method that not only recommends paper and but also summarises the paper based on each sections of the paper and re-ranks the paper based on section wise summary.

5. Methodology

In our proposed methodology, first we have applied the basic pre-processing like removing stopwords, white spaces, etc. Then vectorization of the corpus is done. After that the user query is taken and the pre-processing and the vectorization is applied on it. Once the query vector and the corpus vector are created then the similarity among them is calculated using cosine similarity and the best 'k' research papers are recommended. Based on those recommended paper ids the pdfs are fetched and the content of the pdf is extracted. After that the pre-processing is done on the extracted data to obtain different sections. Once the section is extracted the section wise summary is done and then is combined to form a complete summary. For this summarization part we have used extractive based summary. After that the summary generated is used for again finding the cosine similarity with the paper along with the title and abstract and the paper is re-ranked and recommended. Finally the abstractive summary of the paper that is chosen by user is shown to the user.

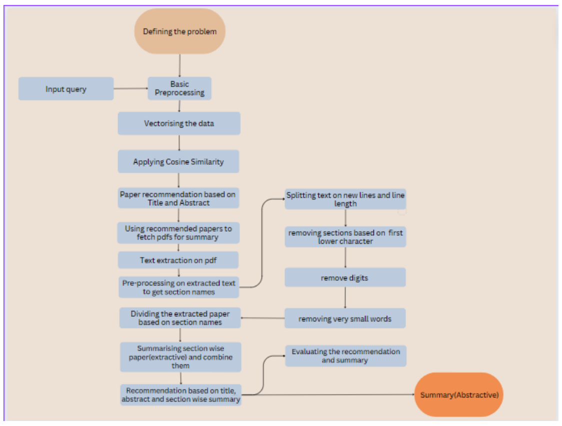

Defining the problem
Input query
Basic Preprocessing
Vectorising the data
Applying Cosine Similarity
Paper recommendation based on Title and Abstract
Using recommended papers to fetch pdfs for summary
Text extraction on pdf
Pre-processing on extracted text to get section names
Dividing the extracted paper based on section names
Summarising section wise paper(extractive) and combine them
Recommendation based on title, abstract and section wise summary
Splitting text on new lines and line length
removing sections based on first lower character
remove digits
removing very small words
Evaluating the recommendation and summary
Summary(Abstractive)

Figure 1 Flowchart of proposed approach

5.1 Pre-processing

5.1.1 Recommendation

For the recommendation part basic pre-processing of NLP like removing of white spaces, removing of stopwords is done.

5.1.2 Text Summarization

For the text summarization first the text which is extracted from pdf is split based on the new line. Then on the split text the section headings are fetched based on the condition if the first character is small or not. Then from that text the text which are digit or float value are removed as they cannot be section headings. After that very small words that either do not have any meaning or cannot be the potential section heading are removed.

5.2 Dataset

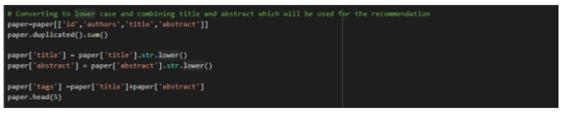
In our work, we have used three different datasets over the course of project. The first dataset is in csv format that contains the title and abstract of the 2000 conference, journal papers and some articles. The dataset was well pre-processed and cleaned but had limited entries.

The second dataset we used was a pdf format dataset that contains 20000 pdfs from the known arxiv dataset having the pdf id associated with each pdf. Since the format of pdfs was too varying and none were following the same pattern thus, recommending the paper based on the extracted data was difficult and computationally heavy. Thus we, have used these pdf extraction approach with our third dataset.

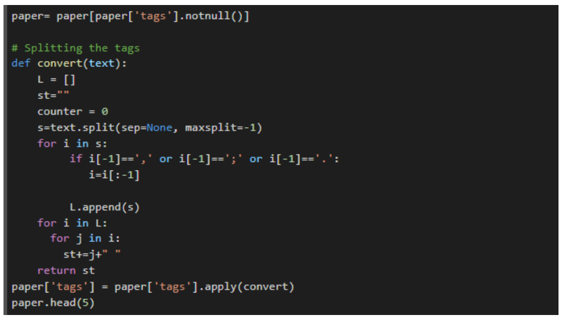
Our third and final dataset is in a json format that contains various keys including title, abstract, authors, etc. and we have used the keys which were apt for our requirement. Using this dataset we have recommended the research papers and based on the recommendation we have used the pdf of the recommended paper to extract the content from it which was further used for re-ranking the recommended papers.

5.3 Code

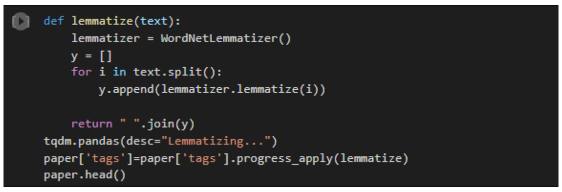
5.3.1 Pre-processing



```
# Converting to lower case and combining title and abstract which will be used for the recommendation
paper=paper[['id','authors','title','abstract']]
paper.duplicated().sum()

paper['title'] = paper['title'].str.lower()
paper['abstract'] = paper['abstract'].str.lower()

paper['tags'] =paper['title']+paper['abstract']
paper.head(5)
```

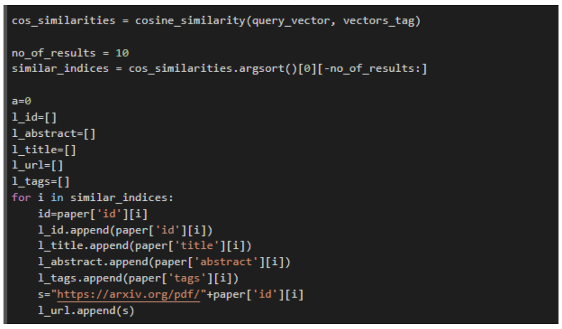


```
paper= paper[paper['tags'].notnull()]

# Splitting the tags
def convert(text):
    L = []
    st=""
    counter = 0
    s=text.split(sep=None, maxsplit=-1)
    for i in s:
         if i[-1]==',' or i[-1]==';' or i[-1]=='.':
            i=i[:-1]

         L.append(s)
    for i in L:
      for j in i:
        st+=j+" "
    return st
paper['tags'] = paper['tags'].apply(convert)
paper.head(5)
```



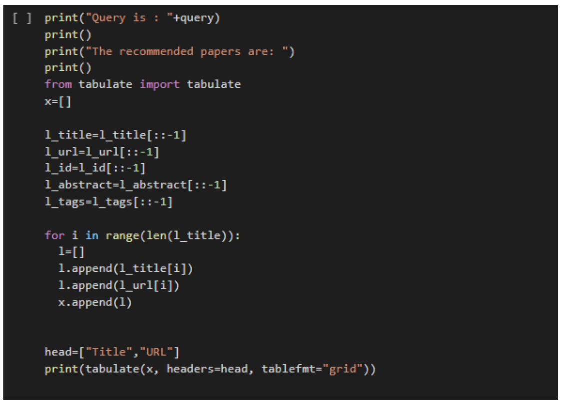
```
def lemmatize(text):
    lemmatizer = WordNetLemmatizer()
    y = []
    for i in text.split():
        y.append(lemmatizer.lemmatize(i))

    return " ".join(y)
tqdm.pandas(desc="Lemmatizing...")
paper['tags']=paper['tags'].progress_apply(lemmatize)
paper.head()
```

5.3.2 Cosine similarity



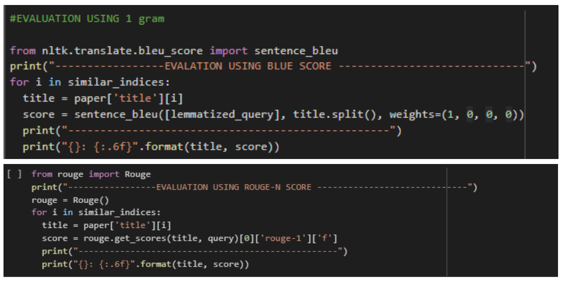
```
cos_similarities = cosine_similarity(query_vector, vectors_tag)

no_of_results = 10
similar_indices = cos_similarities.argsort()[0][-no_of_results:]

a=0
l_id=[]
l_abstract=[]
l_title=[]
l_url=[]
l_tags=[]
for i in similar_indices:
    id=paper['id'][i]
    l_id.append(paper['id'][i])
    l_title.append(paper['title'][i])
    l_abstract.append(paper['abstract'][i])
    l_tags.append(paper['tags'][i])
    s="https://arxiv.org/pdf/"+paper['id'][i]
    l_url.append(s)
```

5.3.2 User input



```
[ ] print("Query is : "+query)
    print()
    print("The recommended papers are: ")
    print()
    from tabulate import tabulate
    x=[]

    l_title=l_title[::-1]
    l_url=l_url[::-1]
    l_id=l_id[::-1]
    l_abstract=l_abstract[::-1]
    l_tags=l_tags[::-1]

    for i in range(len(l_title)):
      l=[]
      l.append(l_title[i])
      l.append(l_url[i])
      x.append(l)


    head=["Title","URL"]
    print(tabulate(x, headers=head, tablefmt="grid"))
```

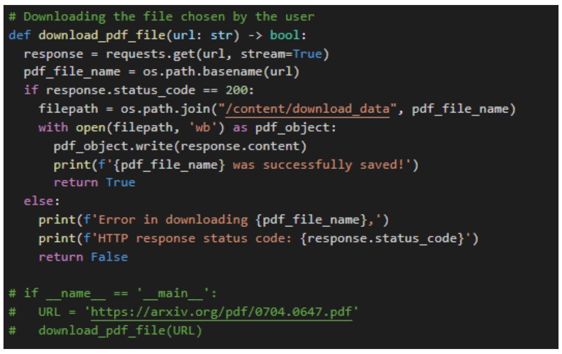
5.3.3 Evaluation



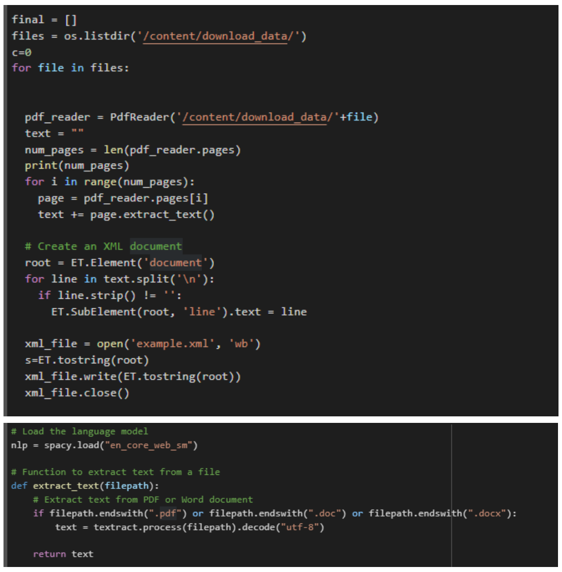
```
#EVALUATION USING 1 gram

from nltk.translate.bleu_score import sentence_bleu
print("-----------------EVALATION USING BLUE SCORE -----------------------------")
for i in similar_indices:
  title = paper['title'][i]
  score = sentence_bleu([lemmatized_query], title.split(), weights=(1, 0, 0, 0))
  print("--------------------------------------------------")
  print("{}: {:.6f}".format(title, score))
```

```
[ ]  from rouge import Rouge
     print("----------------EVALUATION USING ROUGE-N SCORE ----------------------------")
     rouge = Rouge()
     for i in similar_indices:
       title = paper['title'][i]
       score = rouge.get_scores(title, query)[0]['rouge-1']['f']
       print("--------------------------------------------------")
       print("{}: {:.6f}".format(title, score))
```

5.3.4 Summarization



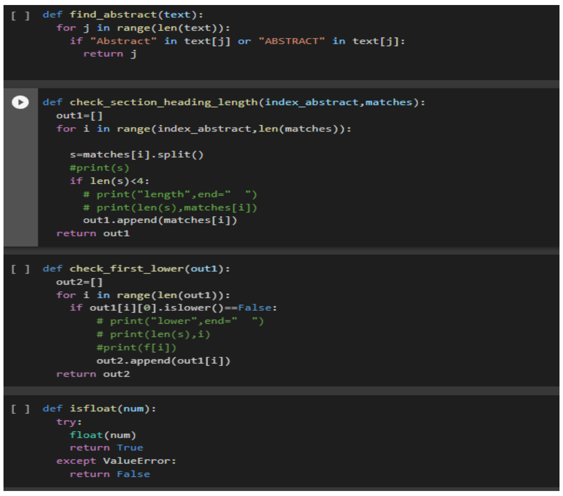
```
# Downloading the file chosen by the user
def download_pdf_file(url: str) -> bool:
    response = requests.get(url, stream=True)
    pdf_file_name = os.path.basename(url)
    if response.status_code == 200:
        filepath = os.path.join("/content/download_data", pdf_file_name)
        with open(filepath, 'wb') as pdf_object:
            pdf_object.write(response.content)
            print(f'{pdf_file_name} was successfully saved!')
            return True
    else:
        print(f'Error in downloading {pdf_file_name},')
        print(f'HTTP response status code: {response.status_code}')
        return False

# if __name__ == '__main__':
#   URL = 'https://arxiv.org/pdf/0704.0647.pdf'
#   download_pdf_file(URL)
```



```
final = []
files = os.listdir('/content/download_data/')
c=0
for file in files:


  pdf_reader = PdfReader('/content/download_data/'+file)
  text = ""
  num_pages = len(pdf_reader.pages)
  print(num_pages)
  for i in range(num_pages):
    page = pdf_reader.pages[i]
    text += page.extract_text()

  # Create an XML document
  root = ET.Element('document')
  for line in text.split('\n'):
    if line.strip() != '':
      ET.SubElement(root, 'line').text = line

  xml_file = open('example.xml', 'wb')
  s=ET.tostring(root)
  xml_file.write(ET.tostring(root))
  xml_file.close()
```

```
# Load the language model
nlp = spacy.load("en_core_web_sm")

# Function to extract text from a file
def extract_text(filepath):
    # Extract text from PDF or Word document
    if filepath.endswith(".pdf") or filepath.endswith(".doc") or filepath.endswith(".docx"):
        text = textract.process(filepath).decode("utf-8")

    return text
```

5.3.5 Summarization pre-processing

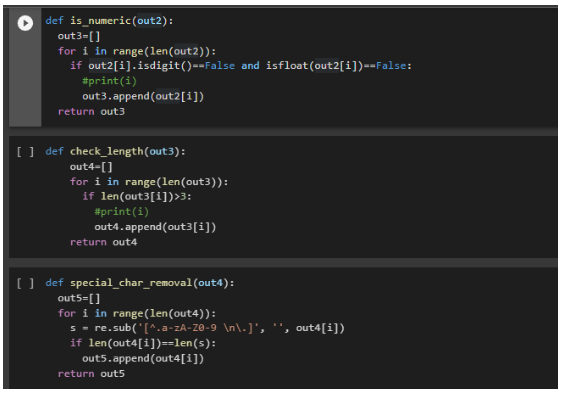


```
[ ]  def find_abstract(text):
       for j in range(len(text)):
         if "Abstract" in text[j] or "ABSTRACT" in text[j]:
           return j
```

```
     def check_section_heading_length(index_abstract,matches):
       out1=[]
       for i in range(index_abstract,len(matches)):

         s=matches[i].split()
         #print(s)
         if len(s)<4:
           # print("length",end="  ")
           # print(len(s),matches[i])
           out1.append(matches[i])
       return out1
```

```
[ ]  def check_first_lower(out1):
       out2=[]
       for i in range(len(out1)):
         if out1[i][0].islower()==False:
             # print("lower",end="  ")
             # print(len(s),i)
             #print(f[i])
             out2.append(out1[i])
       return out2
```

```
[ ]  def isfloat(num):
       try:
         float(num)
         return True
       except ValueError:
         return False
```

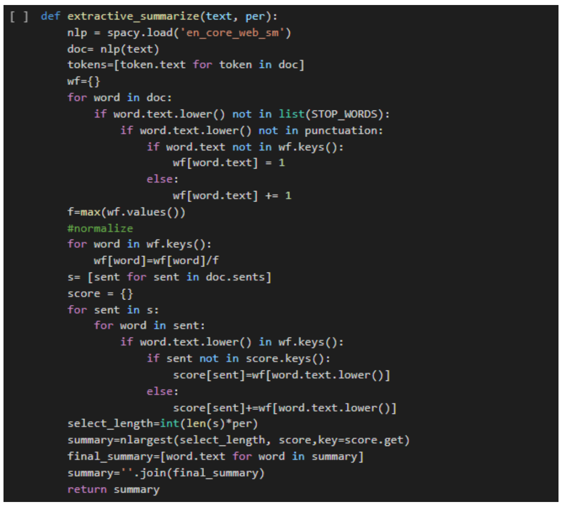


```
     def is_numeric(out2):
       out3=[]
       for i in range(len(out2)):
         if out2[i].isdigit()==False and isfloat(out2[i])==False:
           #print(i)
           out3.append(out2[i])
       return out3
```

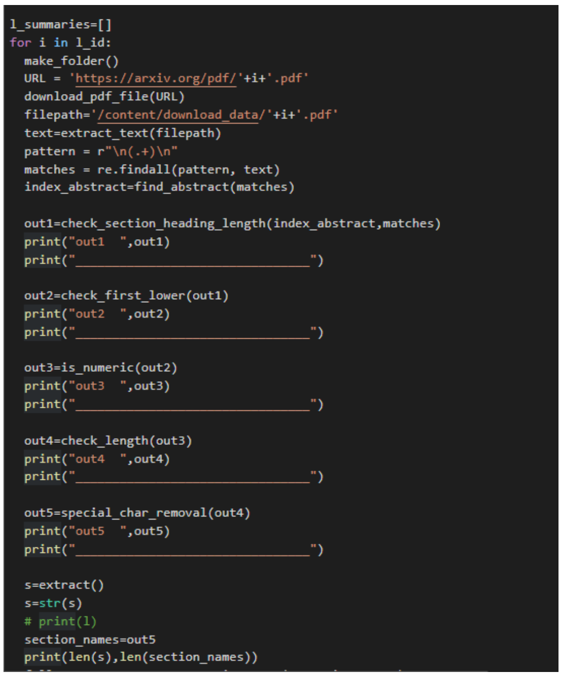
```
[ ]  def check_length(out3):
         out4=[]
         for i in range(len(out3)):
           if len(out3[i])>3:
             #print(i)
             out4.append(out3[i])
         return out4
```

```
[ ]  def special_char_removal(out4):
       out5=[]
       for i in range(len(out4)):
         s = re.sub('[^.a-zA-Z0-9 \n\.]', '', out4[i])
         if len(out4[i])==len(s):
           out5.append(out4[i])
       return out5
```

5.3.5 Extractive summary

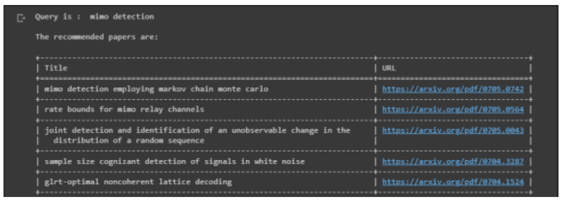


```
[ ]  def extractive_summarize(text, per):
         nlp = spacy.load('en_core_web_sm')
         doc= nlp(text)
         tokens=[token.text for token in doc]
         wf={}
         for word in doc:
             if word.text.lower() not in list(STOP_WORDS):
                 if word.text.lower() not in punctuation:
                     if word.text not in wf.keys():
                         wf[word.text] = 1
                     else:
                         wf[word.text] += 1
         f=max(wf.values())
         #normalize
         for word in wf.keys():
             wf[word]=wf[word]/f
         s= [sent for sent in doc.sents]
         score = {}
         for sent in s:
             for word in sent:
                 if word.text.lower() in wf.keys():
                     if sent not in score.keys():
                         score[sent]=wf[word.text.lower()]
                     else:
                         score[sent]+=wf[word.text.lower()]
         select_length=int(len(s)*per)
         summary=nlargest(select_length, score,key=score.get)
         final_summary=[word.text for word in summary]
         summary=''.join(final_summary)
         return summary
```



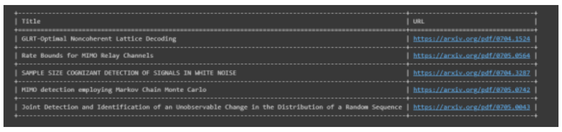
```
l_summaries=[]
for i in l_id:
  make_folder()
  URL = 'https://arxiv.org/pdf/'+i+'.pdf'
  download_pdf_file(URL)
  filepath='/content/download_data/'+i+'.pdf'
  text=extract_text(filepath)
  pattern = r"\n(.+)\n"
  matches = re.findall(pattern, text)
  index_abstract=find_abstract(matches)

  out1=check_section_heading_length(index_abstract,matches)
  print("out1  ",out1)
  print("________________________________________")

  out2=check_first_lower(out1)
  print("out2  ",out2)
  print("________________________________________")

  out3=is_numeric(out2)
  print("out3  ",out3)
  print("________________________________________")

  out4=check_length(out3)
  print("out4  ",out4)
  print("________________________________________")

  out5=special_char_removal(out4)
  print("out5  ",out5)
  print("________________________________________")

  s=extract()
  s=str(s)
  # print(l)
  section_names=out5
  print(len(s),len(section_names))
```

6. Evaluation and Output



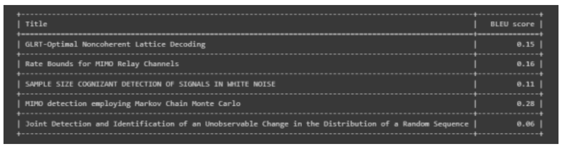
```
Query is :  mimo detection

The recommended papers are:

+----------------------------------------------------------------------+----------------------------------+
| Title                                                                | URL                              |
+======================================================================+==================================+
| mimo detection employing markov chain monte carlo                    | https://arxiv.org/pdf/0705.0742 |
+----------------------------------------------------------------------+----------------------------------+
| rate bounds for mimo relay channels                                  | https://arxiv.org/pdf/0705.0564 |
+----------------------------------------------------------------------+----------------------------------+
| joint detection and identification of an unobservable change in the  | https://arxiv.org/pdf/0705.0043 |
|   distribution of a random sequence                                  |                                  |
+----------------------------------------------------------------------+----------------------------------+
| sample size cognizant detection of signals in white noise            | https://arxiv.org/pdf/0704.3287 |
+----------------------------------------------------------------------+----------------------------------+
| glrt-optimal noncoherent lattice decoding                            | https://arxiv.org/pdf/0704.1524 |
+----------------------------------------------------------------------+----------------------------------+
```

Figure 2 Recommendation without ranking.



```
+--------------------------------------------------------------------------------------+---------------------------------+
| Title                                                                                | URL                             |
+======================================================================================+=================================+
| GLRT-Optimal Noncoherent Lattice Decoding                                            | https://arxiv.org/pdf/0704.1524 |
+--------------------------------------------------------------------------------------+---------------------------------+
| Rate Bounds for MIMO Relay Channels                                                  | https://arxiv.org/pdf/0705.0564 |
+--------------------------------------------------------------------------------------+---------------------------------+
| SAMPLE SIZE COGNIZANT DETECTION OF SIGNALS IN WHITE NOISE                            | https://arxiv.org/pdf/0704.3287 |
+--------------------------------------------------------------------------------------+---------------------------------+
| MIMO detection employing Markov Chain Monte Carlo                                    | https://arxiv.org/pdf/0705.0742 |
+--------------------------------------------------------------------------------------+---------------------------------+
| Joint Detection and Identification of an Unobservable Change in the Distribution of a Random Sequence | https://arxiv.org/pdf/0705.0043 |
+--------------------------------------------------------------------------------------+---------------------------------+
```

Figure 3 Recommendation with ranking.



```
+------------------------------------------------------------------------------------------------------+--------------+
| Title                                                                                                |   BLEU score |
+======================================================================================================+==============+
| GLRT-Optimal Noncoherent Lattice Decoding                                                            |         0.15 |
+------------------------------------------------------------------------------------------------------+--------------+
| Rate Bounds for MIMO Relay Channels                                                                  |         0.16 |
+------------------------------------------------------------------------------------------------------+--------------+
| SAMPLE SIZE COGNIZANT DETECTION OF SIGNALS IN WHITE NOISE                                            |         0.11 |
+------------------------------------------------------------------------------------------------------+--------------+
| MIMO detection employing Markov Chain Monte Carlo                                                    |         0.28 |
+------------------------------------------------------------------------------------------------------+--------------+
| Joint Detection and Identification of an Unobservable Change in the Distribution of a Random Sequence |         0.06 |
+------------------------------------------------------------------------------------------------------+--------------+
```

Figure 4 BLEU score.

7. Conclusion

Although there were systems like Google scholar that recommends the research paper but as the size of information is increasing rapidly the need for the system that summarises the content in order to save the time is important. Thus, we have proposed a system that recommends the paper and generates the summary. The summary is generated section wise and then the re-ranking of the paper is done to provide better results to the user and finally the abstractive summary is provided. Generating a summary is a difficult task as the format of the papers is not uniform also it requires heavy computational power. Thus, we faced these limitations and the system can be improved in future by overcoming these faced challenges.

8. Contribution

8.1 Recommendation

8.1.1 Ranit Pal: Pre-processing of the dataset to clean the dataset. Created the model for cosine similarity and the evaluation of the model. Based on the summarised text re-ranking of recommendation.

8.1.2 Soumyajyoti Das: Analysis of initial dataset and the pre-processing of the dataset and the evaluation and analysis of the different models for the recommendation of research paper.

8.2 Summarization

8.2.1 Chitransh Bose: Analysis of PDFs recommended for the generation of summary. Pre-processing of the analysed data to clean it so that various abstractive models can be implemented on the section wise heading. Implementation of various abstractive approaches and models on the analysed pdfs to create summary that.

8.2.2 Varun Jhunjhunwala: Analysed the pdfs and performed various pre-processing on the data to extract the section wise headings. Implementation of various extractive approaches on the section wise data extracted.

9. References

[1] Beel, Joeran, et al. "Paper recommender systems: a literature survey." *International Journal on Digital Libraries* 17 (2016): 305-338.

[2] Lee, Joonseok, et al. "Local collaborative ranking." *Proceedings of the 23rd international conference on World wide web*. 2014.

[3] Kupiec, Julian, Jan Pedersen, and Francine Chen. "A trainable document summarizer." *Proceedings of the 18th annual international ACM SIGIR*

conference on Research and development in information retrieval. 1995.
[4] Kamruzzaman, S. M., Farhana Haider, and Ahmed Ryadh Hasan. "Text classification using data mining." *arXiv preprint arXiv:1009.4987* (2010).
[5] Yadav, Arun Kumar, et al. "Extractive text summarization using deep learning approach." *International Journal of Information Technology* 14.5 (2022): 2407-2415.