ProfSearch : A Search Engine to Find CS Professors

Akshat Mangal [†*], Gaurav Sonkusle [†*], Mohamed Shamir [†*], Mohmmad Aslam [†], Anirban Dasgupta [†]
Indian Institute of Technology Gandhinagar, India

1 OBJECTIVE

To Build a search engine that can find the most relevant Computer Science Professors working in the field which the user searches for.

2 INTRODUCTION

ProfSearch is a search engine used to search for information about the relevant professor's research area, basic summary and also the rank of professor on the basis of proficiency in their field. To overcome the above problem we have designed our project that searches professors when the user enters the research area and professor name. Our project can be helpful to the researchers and the students who need information about the professors. Using our App they can search the professor and get a summary of the professor.

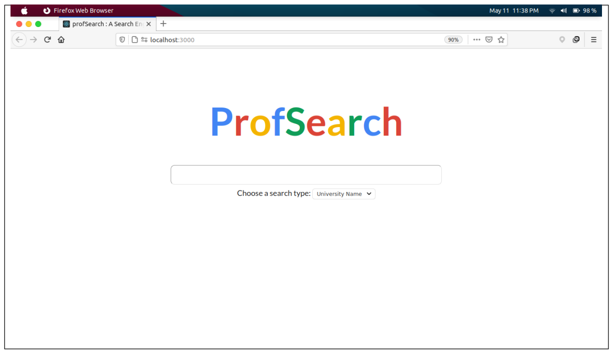


Figure 1: Homepage: Search box

The ProfSearch application is aimed at reducing the time and energy of the students researchers by providing them with the facility of summary and ranking of the professor. On a lighter side, the application is also designed attractively to make it comfortable while using it.

3 FEATURES OF THE PROFSEARCH

- Search By Research Interest
- Search By Professor Name
- Summary Of Each Professor
- Ranking Methods
- Similar Research Interests

[†]{akshat.mangal,gaurav.sonkusle,mohamed.shamir,mohmmad.aslam,anirbandg}@iitgn.ac.in
[*] Undergraduate authors with equal contribution.

4 METHODOLOGY

4.1 Data Collection for the Search Engine

From the website csranking.org, the list of Computer Science Professors can be extracted along with their google scholar id and the links to their homepages. And after getting those data we scrap the google scholar pages using python library named scholarly where as for the summary about the professors we scrap their homepages using pysummarization library in python.

4.1.1 Scholarly.

For scraping the data from the google scholar webpage we used a python library named scholarly which allows us to retrieve author and publication information from google scholar.
Below is a quick demonstrating code for extracting information about the author and their publications using a scholarly library.

```
pip install scholarly

from scholarly import scholarly
search_query = scholarly.search_author('Anirban Dasgupta')
author = next(search_query)

scholarly.pprint(scholarly.fill(author, sections=['basics',
'indices']))
```

Figure 2: Code snippet for Extracting information from Google Scholar

```
{'affiliation': 'Professor, IIT Gandhinagar',
 'citedby': 7752,
 'citedby5y': 3907,
 'email_domain': '@iitgn.ac.in',
 'filled': False,
 'hindex': 29,
 'hindex5y': 23,
 'i10index': 48,
 'i10index5y': 38,
 'interests': ['algorithms', 'data mining', 'social networks'],
 'name': 'Anirban Dasgupta',
 'scholar_id': 'plJC8R0AAAAJ',
 'source': 'SEARCH_AUTHOR_SNIPPETS',
 'url_picture': 'https://scholar.google.com/citations?view_op=me
photo&user=plJC8R0AAAAJ'}
```

Figure 3: Result obtained from the above code

4.1.2 pysummarization - Generating Summary of Professors.

Python's pysummarization library was used to generate summaries from Homepage URLs of the professors. The homepage URLs of the professors were obtained from the database present on the CSRankings Github Repository.

Pysummarization library uses natural language processing and neural network model to summarize the webpage. This library enables us to create a summary with the major points of the original document or web-scraped text that filtered by text clustering. And this library applies accel-brain-base to implement Encoder/Decoder based on LSTM improving the accuracy of summarization by Sequence-to-Sequence(Seq2Seq) learning.
For each URL, the library fetches the summary for the professor and the data is stored in the form of dictionary with "scholarID" of the professor as the keys and the "summaries" as the values. The data from the dictionary was then dumped into json files for proper operation of the web application.

4 shows the snippet of code used to generate some sample summaries from known Homepage URLs. And Figure 5 shows the result, thus, obtained.

4.2 Overall Structure of App

After collecting the data as mentioned in the previous section, we needed to process and store the acquired data in some form of meaningful structure to make proper use of it in the functioning of our application. Hence, we created two databases namely "invertedIndex" and "profData". The schemas for both these databases are described in Figure 7 and Figure 8, while Figure 9 depicts the basic working of our application, the steps taken for each kind-of query are shown in that figure.

As can be seen in the Figure 9, when the user makes a query by research interest, the invertedIndex database is accessed and the application searches for that particular research interest in the invertedIndex, and once found, it returns the list of professors working in the field of that particular interest area and the application displays this resultant information on the Results page in form of cards as programmed in the frontend.

Now, when the query is made by name of the professor, the profData database is accessed and the application searches for that particular name in the profData, and once found, the scholarID of that professor is obtained along with all other relevant details stored in the database and this information is displayed on the Results page by our application.

The third kindof, query can be search by University, in this case also, the profData database is accessed by the application, and it searches for all the professors affiliated to that particular University and returns their information stored in the database, which in turn gets displayed on the Results page by our application.

4.3 Ranking Method

For ranking the professor, we will categorise them using their total citations till 10th May, citations of last five years and h-index. The h-index of the professor tells us about the number of publications and the number of citations per publication that he has. For example, a professor with an h-index of 7 means that the professor has at least seven research papers with a minimum of 7 citations. In

```python
def Main(url):
    # Object of web scraping.
    web_scrape = WebScraping()
    # Web-scraping.
    document = web_scrape.scrape(url)

    # Object of automatic summarization.
    auto_abstractor = AutoAbstractor()
    # Set tokenizer.
    auto_abstractor.tokenizable_doc = SimpleTokenizer()
    # Set delimiter.
    auto_abstractor.delimiter_list = [".", "\n"]
    # Object of abstracting and filtering document.
    abstractable_doc = TopNRankAbstractor()
    # Summarize document.
    result_dict = auto_abstractor.summarize(document, abstractable_doc)

    # Output 5 summarized sentences.
    limit = 5
    i = 1
    summary = ""
    for sentence in result_dict["summarize_result"]:
        summary = summary + " " + sentence
        if i >= limit:
            break
        i += 1
    return summary

scholar_id = ['plJC8R0AAAAJ','U2NUj90AAAAJ']
homepage = ['https://sites.google.com/site/anirbandasgupta/',
            'https://mayank4490.github.io/']
dict_4 = {}
for i in range(2):
  try:
    url = homepage[i]
    summary = Main(url)
    summary = summary.replace("\n","")
    dict_4[scholar_id[i]] = summary

  except:
    dict_4[scholar_id[i]] = ""
    continue
```

Figure 4: Code snippet for Summary Generation

comparison, the i10-index is the number of publications with at least ten citations. Using the three parameters, we calculate the score of each professor using the below formula:

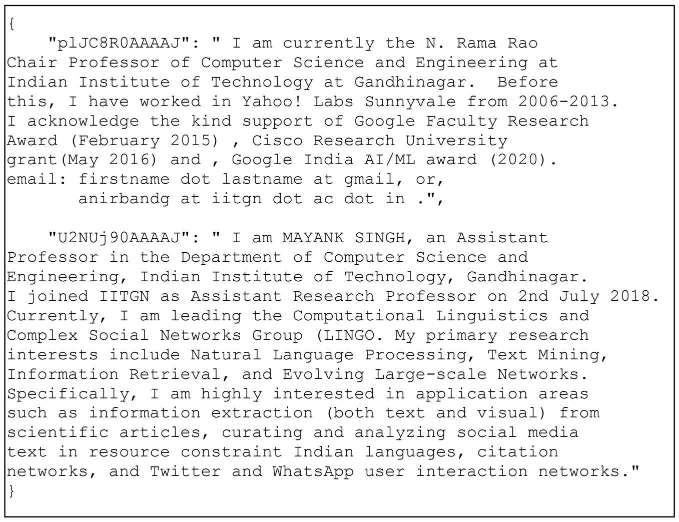
$$a = 520787 \tag{1}$$

$$b = 353903 \tag{2}$$

$$c = 245 \tag{3}$$

$$\alpha = 0.3 \tag{4}$$

$$\beta = 0.2 \tag{5}$$



```
{
    "plJC8R0AAAAJ": " I am currently the N. Rama Rao
Chair Professor of Computer Science and Engineering at
Indian Institute of Technology at Gandhinagar.  Before
this, I have worked in Yahoo! Labs Sunnyvale from 2006-2013.
I acknowledge the kind support of Google Faculty Research
Award (February 2015) , Cisco Research University
grant(May 2016) and , Google India AI/ML award (2020).
email: firstname dot lastname at gmail, or,
       anirbandg at iitgn dot ac dot in .",

    "U2NUj90AAAAJ": " I am MAYANK SINGH, an Assistant
Professor in the Department of Computer Science and
Engineering, Indian Institute of Technology, Gandhinagar.
I joined IITGN as Assistant Research Professor on 2nd July 2018.
Currently, I am leading the Computational Linguistics and
Complex Social Networks Group (LINGO. My primary research
interests include Natural Language Processing, Text Mining,
Information Retrieval, and Evolving Large-scale Networks.
Specifically, I am highly interested in application areas
such as information extraction (both text and visual) from
scientific articles, curating and analyzing social media
text in resource constraint Indian languages, citation
networks, and Twitter and WhatsApp user interaction networks."
}
```

Figure 5: Result obtained from the Code for Summary Generation

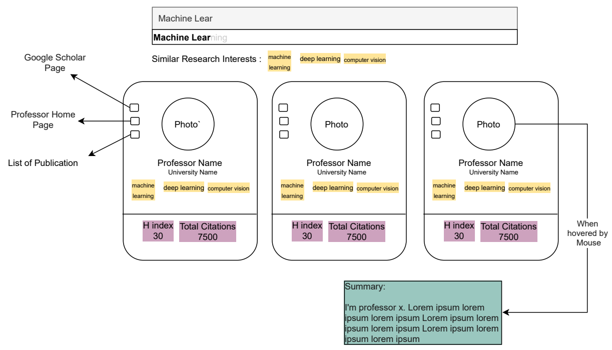
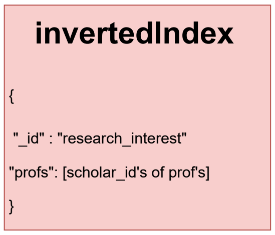


Figure 6: Structure of Results Page



invertedIndex

{

"_id" : "research_interest"

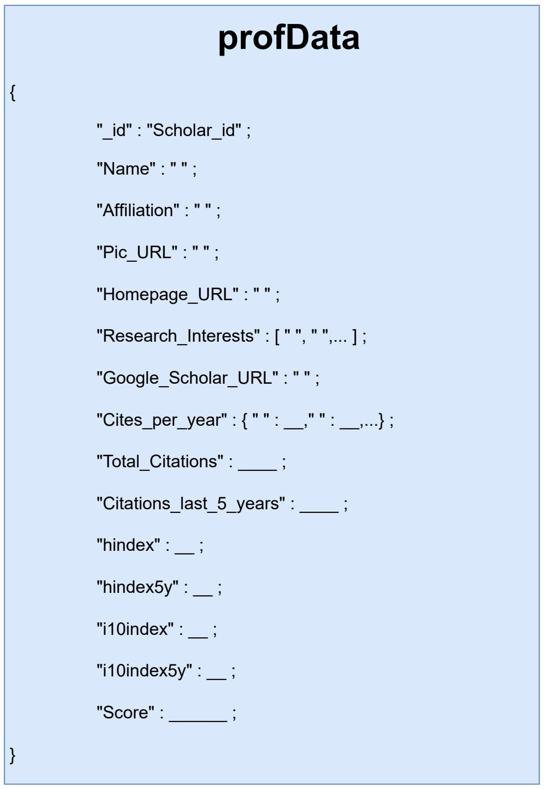
"profs": [scholar_id's of prof's]

}

Figure 7

$$\gamma = 0.5 \tag{6}$$

$$score = \frac{\alpha * (prof_{TotalCitations})}{a} + \frac{\beta * (prof_{CitationsFiveYear})}{b} + \frac{\gamma * (prof_{h-index})}{c} \tag{7}$$



profData

{

"_id" : "Scholar_id" ;

"Name" : " " ;

"Affiliation" : " " ;

"Pic_URL" : " " ;

"Homepage_URL" : " " ;

"Research_Interests" : [" ", " ",...] ;

"Google_Scholar_URL" : " " ;

"Cites_per_year" : { " " : __," " : __,...} ;

"Total_Citations" : ____ ;

"Citations_last_5_years" : ____ ;

"hindex" : __ ;

"hindex5y" : __ ;

"i10index" : __ ;

"i10index5y" : __ ;

"Score" : ______ ;

}

Figure 8

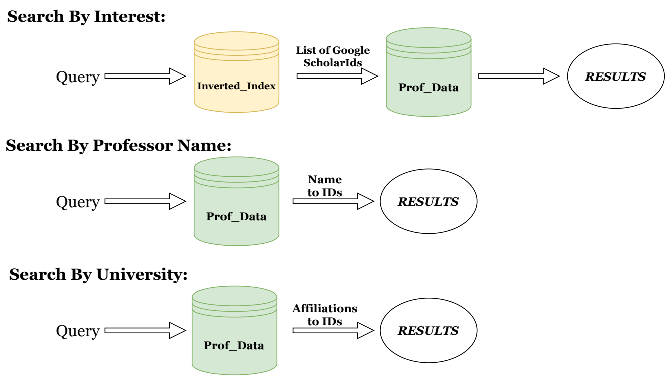
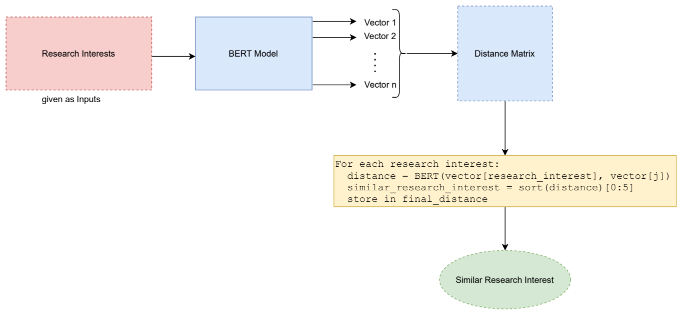


Figure 9

a, b and c represent the maximum value of total citations, citations of the last five year, and h-index present in the dataset. Here, we are assuming the value of alpha, beta, and gamma, but we can train the dataset using machine and deep learning techniques for calculating them in the future.

4.4 Similar Research Interests using BERT

ProfSearch App is used to find professors of some specific research interest however most of the time researchers are also interested in similar research interests to their domain. ProfSearch has a feature to give the list of the five most similar research areas related to the user input. Similar research interests can be obtained using the BERT library. BERT stands for Bidirectional Encoder Representations from Transformers. It is a pre-trained library developed by Google and is based on the transformer architecture. It is pre-trained on large data (around 3,300 million words) of the unlabeled text. Due to the huge amount of data, the BERT have a deeper and intimate understanding of how the language works. It is a "deeply bidirectional" model that means it can learn information from the left and the right side of a sentence. The main things behind the BERT is its Architecture and text preprocessing. The BERT's architecture is set up on top of the Transformer. The BERT has two variants of architecture one is BERT base and the other is BERT large. The flow diagram of BERT implementation is shown in Figure 10.



```
For each research interest:
  distance = BERT(vector[research_interest], vector[j])
  similar_research_interest = sort(distance)[0:5]
  store in final_distance
```

Figure 10: Flow diagram for BERT implementation

The input of the BERT model is the list of all the research interests. The BERT model gives the vector for each of the research interest. These vectors represent the word embedding of the research interest. The distance matrix can be calculated using the spatial distance function. This Distance matrix is the n*n matrix where n is the number of unique research interests. The final output is the Similar Research Interest; this is a dictionary of the dictionary. For each research interest, the distance is the list of all the distance from the others research areas. The similar research interest is the top five closest distance areas.

```
from scipy import spatial
from sent2vec.vectorizer import Vectorizer

final_dic={}
for item1 in interest[10:20]:
  dist_list=[]
  for item2 in interest:
    if item1!=item2:
      query=[item1,item2]
      vectorizer=Vectorizer()
      vectorizer.bert(query)
      vectors_bert = vectorizer.vectors
      dist = spatial.distance.cosine(vectors_bert[0], vectors_bert[1])
  dist_list.append((item2,dist))
  dist_list.sort(key=lambda x:x[1])
  required = dist_list[:5]
  final_dic[item1]=required
```

Figure 11: BERT model code

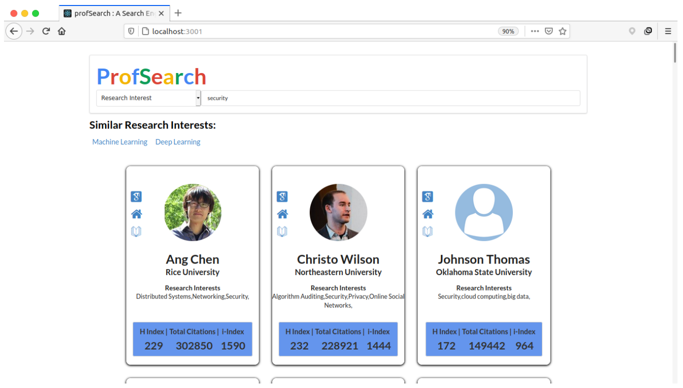
Figure 11 is the main code for the implementation of the BERT model. We use sent2vec library to convert sentence into the vector form. Figure 12 is the output of the above code.

```
{'Algorithms': [('Databases', 0.023767530918121338),
('Embedded', 0.026438951492309570),
('data mining', 0.026924908161163330),
('deep learning', 0.026933312416076660),
('Cryptography', 0.028608262538909912)],
'Cryptography': [('data mining', 0.014752328395843506),
('Artificial intelligence', 0.0157393217086792),
('Computer security', 0.01794302463531494),
('Machine learning', 0.018948733806610107),
('Computer networks', 0.019166648387908936)]}
```

Figure 12: Similar research interest with values

5 DEMONSTRATION

In, the following section you can find the snippets of the working functionality of ProfSearch.



profSearch : A Search En

localhost:3001

ProfSearch

Research Interest | security

Similar Research Interests:

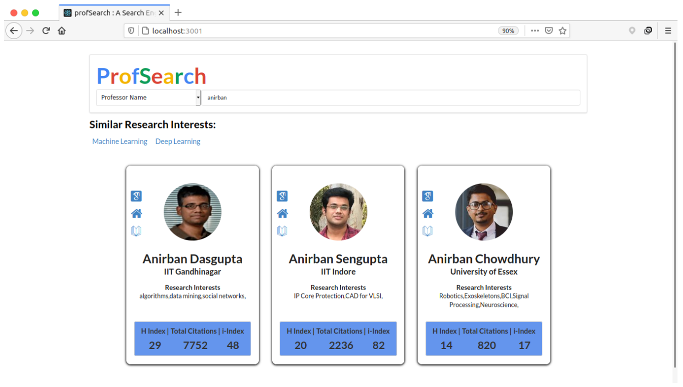
Machine Learning Deep Learning

Ang Chen
Rice University
Research Interests
Distributed Systems,Networking,Security,
H Index | Total Citations | i-Index
229 302850 1590

Christo Wilson
Northeastern University
Research Interests
Algorithm Auditing,Security,Privacy,Online Social Networks,
H Index | Total Citations | i-Index
232 228921 1444

Johnson Thomas
Oklahoma State University
Research Interests
Security,cloud computing,big data,
H Index | Total Citations | i-Index
172 149442 964

Figure 13: Search By Research Interes: Security is given as input.



profSearch : A Search En

localhost:3001

ProfSearch

Professor Name | anirban

Similar Research Interests:

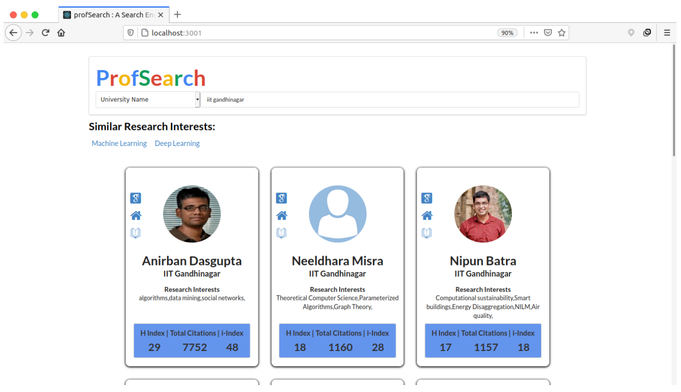
Machine Learning Deep Learning

Anirban Dasgupta
IIT Gandhinagar
Research Interests
algorithms,data mining,social networks,
H Index | Total Citations | i-Index
29 7752 48

Anirban Sengupta
IIT Indore
Research Interests
IP Core Protection,CAD for VLSI,
H Index | Total Citations | i-Index
20 2236 82

Anirban Chowdhury
University of Essex
Research Interests
Robotics,Exoskeletons,BCI,Signal Processing,Neuroscience,
H Index | Total Citations | i-Index
14 820 17

Figure 14: Search By Professor name: Anirban is given as input.



University Name: iit gandhinagar

Similar Research Interests:

Machine Learning Deep Learning

Anirban Dasgupta
IIT Gandhinagar
Research Interests
algorithms,data mining,social networks,

H Index	Total Citations	i-Index
29	7752	48

Neeldhara Misra
IIT Gandhinagar
Research Interests
Theoretical Computer Science,Parameterized Algorithms,Graph Theory,

H Index	Total Citations	i-Index
18	1160	28

Nipun Batra
IIT Gandhinagar
Research Interests
Computational sustainability,Smart buildings,Energy Disaggregation,NILM,Air quality,

H Index	Total Citations	i-Index
17	1157	18

Figure 15: Search By University: IIT Gandhinagar is given as input.

6 CHALLENGES

(1) **Massive Amount of Data:** Although, we initially started with around 25000 professors, but after some processing, we were left with 12500 professors. For these 12500 professors, we have created our databases(,i.e., invertedIndex and profData) and the data extraction process for very cumbersome. We adapted techniques to run the programs parallely on different devices for different sets of data to extract the information and finally, merged all the data into one big dataset. This was done for both the tasks, while extracting information from scholar pages using the scholarly library and also, while generating summaries of the professors using the homepage URLs using the pysummarization library.

(2) **Storing in Data Structures:** Such large size of data cannot be conveniently stored in data structures like lists, tuples, etc. So, figuring out which data structure to use for storing the data and then, processing it to create a database which can be accessed by our application was also a challenging task.

(3) **BERT Problem:** One query to find the distance between two sentence in the BERT takes around 0.5 second to complete in our system. And we have the around 13000 unique research areas in our database. To run all the query, the code will run for 13000X13000 times. So the overall time it take will be huge. So we took the most 100 used research interest for now to get the result of similar research area.

(4) **Troubles with Frontend Backend:** Implementing both the backend and frontend for the proper functioning of our application was also a difficult task. A lot of bugs and errors were encountered.

7 FUTURE WORK

(1) **Autocompletion:** When a user types the search input, it autocompletes the word.

(2) **Error Detection:** Spelling correction (A feature similar to “showing results for” in Google Search).

8 GITHUB REPOSITORY

The codes for the implementations is available at the github repository **github.com/mshamir11/profSearch**.

9 REFERENCE

- Rizvi, M. S. Z. (2019). Demystifying BERT: A Comprehensive Guide to the Groundbreaking NLP Framework.
- https://scholarly.readthedocs.io/en/stable/quickstart.html
- Berger, E. (2018). CSRankings: Computer Science Rankings.
- What Is the Difference between H-Index, I10-Index, and G-Index?” ResearchGate, ResearchGate, 21 Dec. 2012,
- Summarization :https://pypi.org/project/pysummarization/
- BERT Vectors : https://pypi.org/project/sent2vec/