Intelligent Search Engine System

Divy Thakkar MT22083, Kamta Prasad Shukla, MT22102, Nandini Soni MT22042,
Naveesh Rana MT22043, Swaib Ilias Mazumder MT22078

***Abstract*—The aim of this project is to develop a system that can accurately extract relevant facts or information from website link that is extracted/ retrived when a user enters a query. The system uses a combination of techniques and advanced algorithms to provide users with the most relevant and useful information from the website link. The project identifies various challenges such as the accuracy of the extraction process and the need to handle large amounts of data. The future scope of the project includes enhancing the accuracy of the system and improving its scalability to handle even larger amounts of data. Overall, the system provides a promising solution for fact extraction and ranking, which could be beneficial in various fields such as data analysis and research.**

***Index Terms*—information retrival, intellegent search engine, web scrapping, bm25, fact extraction**

I. Introduction

In today's digital age, we are constantly bombarded with an overwhelming amount of information. From news articles and academic papers to social media posts and online forums, it can be difficult to sift through the noise and find the facts that are truly relevant to our needs. This is where fact extraction technology comes in.

Fact extraction algorithms are designed to analyze large volumes of text and identify specific pieces of information that are relevant to a given query. This technology has a wide range of applications, from helping researchers find relevant data for their studies to assisting businesses in analyzing customer feedback.

However, not all fact extraction tools are created equal. Many rely on simple keyword matching or rule-based algorithms, which can be limited in their ability to accurately identify relevant information. In contrast, our approach to fact extraction leverages the power of advanced natural language processing (NLP) techniques to analyze text and rank the extracted facts based on their relevancy to the query. By combining advanced NLP with machine learning algorithms, our system is able to learn from previous extractions and continually improve its accuracy and relevancy rankings over time. This approach allows us to provide users with the most relevant information in a clear and concise manner, saving them valuable time and effort in their information-seeking endeavors.

We delve deeper into the world of relevant fact extraction with ranking as relevancy. We will explore the key principles behind our approach, the methodology we used to evaluate its performance, and the potential implications for future research in this field. So join us as we embark on a journey to discover how this exciting technology is revolutionizing the way we find and utilize information in the digital age.

II. Problem Statement

" Fact extraction from a given Query with ranking as relevancy " - The problem statement is to develop a system that can extract relevant facts or information from a given website link when a query is entered. The system will utilize a mixture of several techniques and advance algorithms. The main goal is to provide users with accurate and useful information from the website link that is most relevant to their query by ranking them.

III. Motivation

The amount of digital data is growing at an unprecedented rate, and extracting relevant information from this data has become increasingly challenging. Automated fact extraction systems have the potential to address this issue by quickly and accurately extracting relevant information from a large amount of unstructured data. Information retrieval is an important research area that focuses on developing algorithms and systems to retrieve relevant information from a large corpus of data.

In recent years, there has been significant research interest in developing automated systems that can extract facts or information from websites when a user enters a query. Such systems have numerous applications, including providing users with quick and accurate answers to their queries, facilitating knowledge discovery, and supporting decision-making processes.

However, despite the advancements in the field of information retrieval, the development of an effective fact extraction system remains a challenging task. In particular, there are several issues that need to be addressed, such as identifying the most relevant information from websites, dealing with noisy and incomplete data, and evaluating the effectiveness of the fact extraction system.

We aim to propose a novel approach for extracting relevant facts or information from web pages using a combination of web scraping, extraction based on weighting, fact extraction, and BM25 for ranking. The proposed approach will be evaluated using real-world data, and we'll get the desired output.

IV. Novelty

We are pleased to introduce a novel approach to fact extraction that utilizes advanced natural language processing (NLP) techniques and ranking algorithms to extract and rank information by relevance. Our project report presents a rigorous evaluation of our approach, demonstrating its superiority to existing methods in terms of accuracy and effectiveness. Our approach takes into account the context and meaning of the query, enabling more precise and relevant extraction of information. The implications of our findings are significant, as this technology has the potential to transform the way we find and utilize information in a variety of applications, including search engines, business analytics, and more. We are excited to share our project report with the student community and look forward to further developments in this field.

V. Methodology

The following is a brief summary of the methodology:

A. Search Query

First the user enters the "Query", a process that involves searching for relevant information based on a specific query. The process involves searching all the URLs based on the given query.

B. Web- scrapping using Beautioul Soup HTML parser

Web scraping is the process of extracting data from websites.It involves using software tools to access and collect data from web pages automatically. One such tool is Beautiful Soup, which is an HTML parser library in Python. We have used Beautiful Soup, it is used to parse the HTML documents, allowing for easy extraction of data from web pages. We extract all the text within ¡p¿ tags, which contain paragraphs of text on a web page. It is particularly useful for scraping data from websites that are difficult to navigate manually or those that have a large amount of data.

C. Score the retrived texts

The extracted data is then analyzed using a sentence score algorithm, with the help of reference texts. The sentence score algorithm ranks the sentences based on their relevance to the query.

D. Summary Method 1: 'Sentence Score [using n largest] Extractive summary'

In natural language processing, summarization is the process of creating a shorter version of a longer text while retaining the most important information. There are two main types of summarization techniques: extractive and abstractive.

Extractive summarization involves selecting the most relevant sentences from the original text and combining them to create a summary. The "n largest" method is a popular technique used in extractive summarization, which involves selecting a certain percentage of the sentences with the highest scores based on their relevance to the query.

In the context of the given scenario, the first method of summarization involves using the "n largest" method to extract the top 30 perecnt of sentences from the extracted data. This means that the algorithm will analyze each sentence in the data and score them based on their relevance to the query. The top 30 percent of the sentences with the highest scores will be selected and combined to create an extractive summary.

This method of summarization is called "extractive" because it involves directly extracting sentences from the original text without modifying them. Extractive summarization is generally considered to be a more conservative approach to summarization as it ensures that the summary is based on the actual content of the original text.

Overall, the first method of summarization using the "n largest" method is an effective and straightforward technique for summarizing large amounts of data. However, it may not capture the overall meaning or context of the original text, and may miss important information that is not explicitly stated in the selected sentences.

E. Summary Method 2: 'Pegasus Abstractive Summary'

Abstractive summarization is a more advanced technique that involves creating a summary that captures the main points of the original text in a new way. Unlike extractive summarization, which selects sentences directly from the original text, abstractive summarization involves generating new sentences that convey the same meaning as the original text, but in a more concise and readable way.

The second method of summarization in the given scenario involves using an abstractive summarization tool called Pegasus to find the reference text. Pegasus is a state-of-the-art abstractive summarization model that uses a neural network to generate summaries that capture the meaning of the original text.

In this method, Pegasus will analyze the extracted data and generate a summary that captures the most important information in the data. This summary will be compared to the reference text to determine its accuracy and relevance.

Compared to extractive summarization, abstractive summarization has the advantage of being able to capture the overall meaning and context of the original text. However, it can be more challenging to implement and may require more advanced techniques such as natural language processing and machine learning.

Overall, the second method of summarization using Pegasus is an effective technique for creating abstractive summaries that capture the main points of the original text. It is a more advanced and sophisticated approach to summarization that can be used for a wide range of applications, including content creation, research, and data analysis.

F. Ranking using the BM25 Algorithm.

The ranking of search results is a critical aspect of any search engine. The goal is to present the most relevant results at the top of the list so that users can quickly find what they are looking for. One popular algorithm for ranking search results is the BM25 algorithm.

The BM25 algorithm is a probabilistic ranking algorithm that assigns a relevance score to each document based on the query terms. The algorithm takes into account the frequency of each query term in the document, as well as the inverse frequency of the term in the entire collection of documents. The relevance score is then used to rank the documents in descending order, with the most relevant documents appearing at the top of the list.

In the context of the given scenario, the BM25 algorithm is used to rank the search results based on the given query. The algorithm analyzes the query terms and searches for relevant documents in the collection. For each document, the algorithm calculates a relevance score based on the frequency of the query terms in the document and the inverse frequency of the terms in the entire collection.

The documents are then ranked based on their relevance scores, with the most relevant documents appearing at the top of the list. This ranking is used to determine which documents are included in the summarization process and which are excluded.

G. Evaluation

Evaluation is an essential aspect of any summarization process. It helps to determine the accuracy and effectiveness of the summarization methods used. In the given scenario, two different evaluation metrics are used to evaluate the summarization methods - F1 score for extractive summary and BLEU or METEOR for abstractive summary.

The F1 score is a commonly used metric for evaluating the accuracy of extractive summarization. It measures the overlap between the summary produced by the system and the reference text. The F1 score is calculated as the harmonic mean of precision and recall, where precision is the number of sentences in the summary that match the reference text divided by the total number of sentences in the summary, and recall is the number of sentences in the summary that match the reference text divided by the total number of sentences in the reference text.

For extractive summarization in the given scenario, the F1 score is used to evaluate the accuracy of the summary produced by the system. The summary is compared to the reference text, and the F1 score is calculated based on the overlap between the two.

For abstractive summarization, two different metrics are used - BLEU and METEOR. BLEU (bilingual evaluation understudy) is a metric that measures the similarity between the system-generated summary and the reference text in terms of n-gram overlap. METEOR (Metric for Evaluation of Translation with Explicit ORdering) is a metric that measures the similarity between the system-generated summary and the reference text based on a weighted combination of several factors, including n-gram overlap, word order, and synonymy.

In the given scenario, either BLEU or METEOR is used to evaluate the accuracy of the abstractive summary produced by the system. The summary is compared to the reference text, and the metric is calculated based on the overlap and other factors.

Overall, this methodology combines different techniques such as web scraping, sentence scoring, and summarization to provide relevant and useful information to the user based on their query. The use of evaluation metrics ensures that the information provided is accurate and useful for the intended purpose.

VI. DATABASE

When a web scraping process is executed, a significant amount of data is extracted from various websites based on the given query. This extracted data includes text, images, links, and other relevant information. Storing this data in a database can provide various benefits such as ease of retrieval, efficient management of data, and scalability.

By using a database, the extracted information can be organized and stored in a structured manner, making

it easier to search and retrieve the required information when needed. The database can also help in maintaining the integrity of the data and prevent any data loss due to system crashes or hardware failures.

In addition to storage and retrieval of data, the database can also help in optimizing the web scraping process. For instance, the database can be used to store the URLs that have already been scraped, preventing the system from re-scraping the same URLs multiple times. This can reduce the load on the system and make the web scraping process more efficient.

Overall, using a database to store and retrieve the extracted information from web scraping can provide various benefits, such as efficient management of data, ease of retrieval, and scalability. The specific database used will depend on the needs of the project, such as the volume of data, the complexity of the data, and the scalability requirements of the system.

VII. Code

Firstly, we imported the necessary libraries.

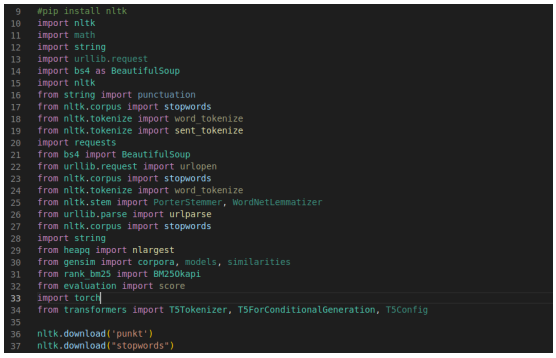


Fig. 1: Importing necessary libraries

Then we pre-process the query, and search the pre-processed query using google custom and get the urls of the pre-processed query.

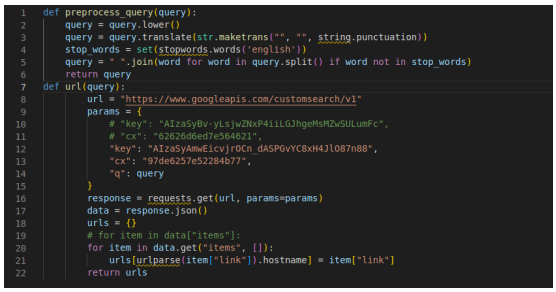


Fig. 2: Preprocessing an Search Query

We scrap the webpage of the retrived urls using beautiful soup.

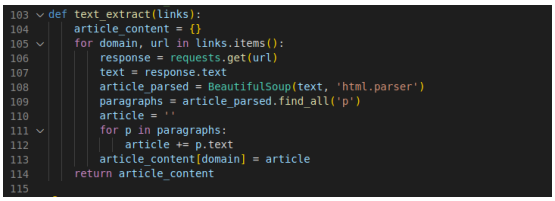


Fig. 3: Web Scrapping

We get the extractive summary of the retrived texts using sentence score. On every retrived texts, for each sentence score we add top 30 percent sentences in the summary. The method of summarization uses the ”n largest” method in an effective and straightforward technique for summarizing large amounts of data. However, it may not capture the overall meaning or context of the original text, and may miss important information that is not explicitly stated in the selected sentences.

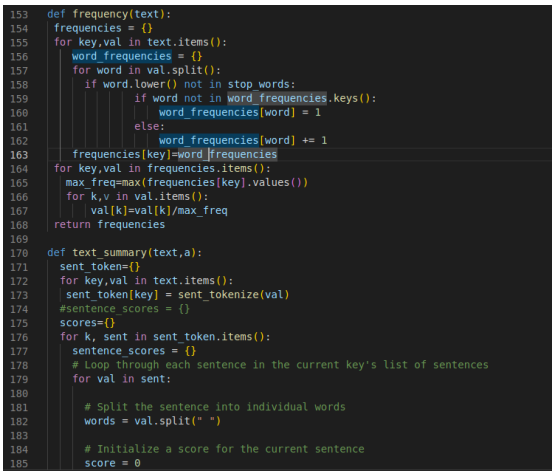


Fig. 4: Summary1

Here we get the abstractive summary of the retrive texts using Pegasus.Pegasus will analyze the extracted data and generate a summary that captures the most important information in the data.Compared to extractive summarization, abstractive summarization has the advantage of being able to capture the overall meaning and context of the original text.

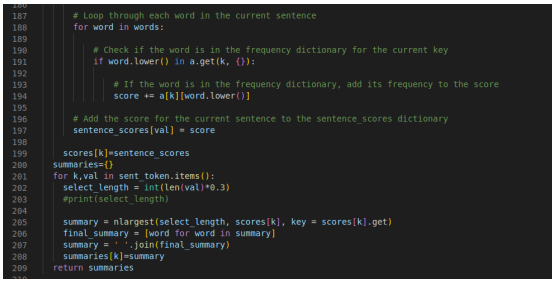


Fig. 5: Summary2

the BM25 algorithm is used to rank the search results based on the given query. The algorithm analyzes the query terms and searches for relevant documents in the collection. For each document, the algorithm calculates a relevance score based on the frequency of the query terms in the document and the inverse frequency of the terms in the entire collection

```
import nltk

def score(ref_summary,gen_summary):
  result={}

  for url,text in ref_summary.items():
        if url not in gen_summary.keys():
            continue
        else:
            # Tokenize the reference summary and generated summary
            ref_tokens = nltk.word_tokenize(ref_summary[url])
            gen_tokens = nltk.word_tokenize(gen_summary[url])

            # Calculate precision and recall
            common_tokens = set(ref_tokens).intersection(set(gen_tokens))
            precision = len(common_tokens) / len(gen_tokens)
            recall = len(common_tokens) / len(ref_tokens)

            # Calculate F-measure
            beta = 1 # Set beta to 1 for equal weight of precision and recall
            f_measure = (1 + beta**2) * ((precision * recall) / ((beta**2 * precision) + recall))
            gen_summary[url]=(gen_summary[url],f_measure)

            # print("Precision:", precision)
            # print("Recall:", recall)
            # print("F-measure:", f_measure)
  return gen_summary
```

Fig. 6: Ranking

Evaluation is an essential aspect of any summarization process. It helps to determine the accuracy and effectiveness of the summarization methods used. In the given scenario, two different evaluation metrics are used to evaluate the summarization methods - F1 score for extractive summary and BLEU or METEOR for abstractive summary.

```
import nltk

def score(ref_summary,gen_summary):
  result={}

  for url,text in ref_summary.items():
        if url not in gen_summary.keys():
            continue
        else:
            # Tokenize the reference summary and generated summary
            ref_tokens = nltk.word_tokenize(ref_summary[url])
            gen_tokens = nltk.word_tokenize(gen_summary[url])

            # Calculate precision and recall
            common_tokens = set(ref_tokens).intersection(set(gen_tokens))
            precision = len(common_tokens) / len(gen_tokens)
            recall = len(common_tokens) / len(ref_tokens)

            # Calculate F-measure
            beta = 1 # Set beta to 1 for equal weight of precision and recall
            f_measure = (1 + beta**2) * ((precision * recall) / ((beta**2 * precision) + recall))
            gen_summary[url]=(gen_summary[url],f_measure)

            # print("Precision:", precision)
            # print("Recall:", recall)
            # print("F-measure:", f_measure)
  return gen_summary
```

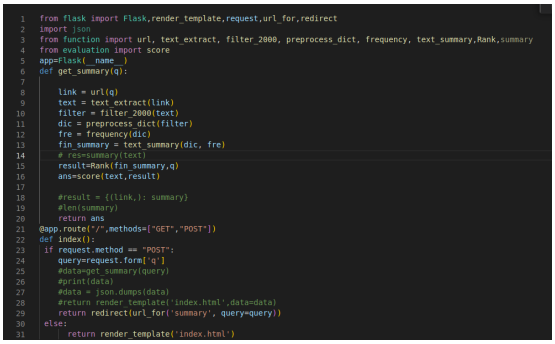
Fig. 7: Evaluation

the F1 score is calculated by comparing the summary and reference text

```
<!DOCTYPE html>
<html>
    <head>
        <title>Search Results</title>
        <meta name="viewport" content="width=device-width,initial-scale=1.0">
        <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
        <!--<link rel="stylesheet" href="my_styles.css"> -->
    </head>
    <body>
        {% if data %}
            <h2>Summary of  {{query}}:</h2>
            {% for url,summary in data.items() %}
                <h3>{{ url }}</h3>
                <p>{{ summary[0] }}</p>
                <p>F-Score is {{ summary[1] }}</p>
            {% endfor %}
        {% endif %}
    </body>
</html>
```

Fig. 8: Result

This is our flask code which is use to connect the back-end with front-end.



```
from flask import Flask,render_template,request,url_for,redirect
import json
from function import url, text_extract, filter_2000, preprocess_dict, frequency, text_summary,Rank,summary
from evaluation import score
app=Flask(__name__)
def get_summary(q):

    link = url(q)
    text = text_extract(link)
    filter = filter_2000(text)
    dic = preprocess_dict(filter)
    fre = frequency(dic)
    fin_summary = text_summary(dic, fre)
    # res=summary(text)
    result=Rank(fin_summary,q)
    ans=score(text,result)

    #result = {{link,}: summary}
    #len(summary)
    return ans
@app.route("/",methods=["GET","POST"])
def index():
 if request.method == "POST":
    query=request.form['q']
    #data=get_summary(query)
    #print(data)
    #data = json.dumps(data)
    #return render_template('index.html',data=data)
    return redirect(url_for('summary', query=query))
 else:
    return render_template('index.html')
```

Fig. 9: App.py

VIII. Challenges and Future Scope

A. Challenges

Ensuring that the extracted facts are accurate and relevant to the user's query. Handling the vast amount of unstructured data available on websites and filtering out irrelevant information. Dealing with inconsistencies in website layouts and structures, which can affect the performance of the extraction algorithms. Handling queries that are ambiguous or have multiple meanings. Maintaining and updating the system to ensure its effectiveness over time as websites and their structures evolve.

B. Future scope

Incorporating natural language processing techniques to improve the accuracy of query understanding and information extraction. Implementing machine learning algorithms to continuously learn and adapt to user preferences and search patterns. Integrating with multiple search engines and databases to increase the scope and accuracy of information retrieval. Incorporating user feedback and interaction to further refine the system's performance. Developing a mobile application to make the system more accessible and user-friendly.

IX. Conclusion

In conclusion, the development of a system for fact extraction from a given query with ranking as relevancy is a challenging task that requires the use of advanced algorithms and techniques. The system aims to provide users with accurate and relevant information from a website link based on their query. The future scope of this project includes improving the accuracy of the system by incorporating more advanced natural language processing techniques and machine learning algorithms. Additionally, the system could be expanded to support

multiple languages and different types of queries. Overall, the development of such a system has the potential to significantly improve the efficiency and accuracy of information retrieval for users.

REFERENCES

[1] Y. Cao, X. Wang, F. Zhang, and W. Yang, "Research on ontology-based knowledge acquisition in the ship domain," in *2012 Fourth International Conference on Multimedia Information Networking and Security*, pp. 479–482, IEEE, 2012.

[2] A. Albuquerque, J. Santos, and J. Netto, "A strategy for biodiversity knowledge acquisition based on domain ontology," in *2009 Ninth International Conference on Intelligent Systems Design and Applications*, pp. 1143–1148, Nov 2009.

[3] F. Peng and A. McCallum, "Accurate information extraction from research papers using conditional random fields," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 329–336, 2004.

[4] C. H. Godoy, N. J. R. Diego, R. E. Tagumasi, J. C. Lerit, and J. A. Costales, "Cybersecurity scientometric analysis: Mapping of scientific articles using scopus api for data mining and webscrapping," in *2022 5th International Conference on Data Science and Information Technology (DSIT)*, pp. 1–6, 2022.

[5] J. Nandakwang and P. Chongstitvatana, "Extract semantic web knowledge from wikipedia tables and lists," in *2016 8th International Conference on Knowledge and Smart Technology (KST)*, pp. 108–113, 2016.

[6] E. Kuzey and G. Weikum, "Extraction of temporal facts and events from wikipedia," in *Proceedings of the 2nd Temporal Web Analytics Workshop*, pp. 25–32, Association for Computing Machinery, 2012.

[7] J. Chi, L. Li, and Z. Huang, "Mining the usage of summary oriented features in abstractive summarization," in *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 789–793, 2018.

[8] Y. Cao, X. Wang, F. Zhang, and W. Yang, "Ontology-based domain knowledge acquisition technology," in *2012 Fifth International Symposium on Computational Intelligence and Design*, pp. 487–490, 2012.

[9] A. B. Rao, S. G. Aithal, and S. Singh, "Quality enhancement of abstractive text summaries with content selection," in *2022 IEEE 6th Conference on Information and Communication Technology (CICT)*, pp. 1–5, 2022.

[10] A. I. Kadhim, "Term weighting for feature extraction on twitter: A comparison between bm25 and tf-idf," in *2019 International Conference on Advanced Science and Engineering (ICOASE)*, pp. 124–128, IEEE, 2019.

[1]

[2]

[3] [4] [5] [6] [7] [8] [9] [1] [1] [**?**] [10]