PROJECT FINAL REPORT

Project Name	Medical Information Retrieval and Management
Submission Date	4/20/2023
Prepared by	Group 11

Team Members

Team Member	Phone	Email	Notes
Chenchu Siva Koushik Vemula	9409774833	ChenchuSivaKoushikVemula@my.unt.edu	
Poojitha Navuluru	9408433067	PoojithaNavuluru@my.unt.edu	
Nikhil Konduru	9403154277	NikhilKonduru@my.unt.edu	

Workflow

Action Items

Task or Deliverable	Task Owner
Identifying the dataset	Nikhil
Analyzing the dataset	Koushik
Loading and cleaning the dataset	Poojitha
Constructing Inverted Index	Koushik
Calculating TF-IDF	Poojitha
Calculation of Cosine similarity	Koushik
Implementing Document Summarization	Poojitha
Evaluation of Document Summarization	Nikhil
Final Report Creation	Poojitha,Nikhil,Koushik
Frontend Implementation	Koushik

Abstract

A critical component of healthcare systems is the management and retrieval of medical information. Managing and retrieving information has become a difficult undertaking as a result of the quick expansion of medical information. Medical personnel can access and recover patient information more quickly and conveniently with the use of a dependable medical information management and retrieval system, improving patient care. The purpose of this paper is to provide an overview of the idea of managing and retrieving medical information, the significance of doing so for healthcare, and the numerous approaches and technologies that can be used to do so. Healthcare professionals can more easily and rapidly access the relevant data by retrieving and summarizing medical reports, which will improve patient care.

Searching for and obtaining pertinent medical information from a variety of sources, including medical databases, electronic health records, and medical literature, is known as medical information retrieval. Finding medical records can take a while, especially when there is a lot of information involved. Medical information retrieval systems utilize machine learning and natural language processing to help healthcare providers find and quickly retrieve pertinent medical information.

Medical report summarizing is the creation of brief, intelligible summaries from medical records. Because medical reports are sometimes lengthy and filled with technical jargon, it can be challenging for healthcare professionals to immediately identify the pertinent information. Medical report summarizing systems collect the most crucial information from medical reports and condense it using machine learning algorithms and methods of natural language processing.

Project Design

Information Retrieval (IR): IR is a design technique used to look for pertinent data in huge datasets. It is employed in medical information retrieval to extract pertinent patient data from clinical trial datasets, medical literature databases, and electronic health record systems.

Natural Language Processing (NLP): NLP is a branch of artificial intelligence that focuses on processing and analyzing human language. It is a fundamental technology used in medical information retrieval and summarization of reports.

Document search

To retrieve the medical records from corpus we are using TF-IDF method to effectively retrieve a record based on user query. The acronym "Term Frequency – Inverse Document Frequency" stands for this. This method counts the number of words in a collection of documents. Each word is typically given a score to indicate how important it is to the document and corpus. Information retrieval and text mining applications frequently use this method.

Any computer language can more easily comprehend textual data when it is presented as a numerical value. Therefore, in order to better represent the text, we must vectorize all of it.We may further carry out numerous activities, such as locating the pertinent papers, rating, clustering, etc. by vectorizing the documents. When you conduct a google search, this specific method is employed.

TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF)

- t – term (token)
- d – document (set of tokens)
- N – count of documents
- corpus – the entire set of documents

Term Frequency

The frequency of a word in a document depends on the length of the document and the generality of the word. To normalize the frequency value, we divide the frequency with the total number of words in the document.

tf(t,d) = count of t in d / number of words in d

Document Frequency

The most important details are that TF is the frequency counter for a term in document d, while DF is the count of occurrences of term t in the document set N.

df(t) = occurrence of t in N documents

Inverse Document Frequency

IDF is the inverse of document frequency, which measures the informativeness of a term. It is low for the most frequent words, such as stop words, as they are present in almost all of the documents. This gives a relative weightage to the word.

$$\text{tf-idf}(t, d) = \text{tf}(t, d) * \log(N/(df + 1))$$

Implementation

Dataset Analysis

Analyzing the data is the first step in any machine learning activity. The dataset comprises of two folders containing English papers with various names. Different alignment patterns can be used to distinguish the title, however the majority are center aligned. Each folder contains an index.html file that may be used to browse the dataset. This file lists all the document names and their titles. In other words, the titles are provided to the user without being fully extracted from each document.

Loading and cleaning the dataset

Here we load text files into database for further searching the documents. Before any processing is done the data needs to be cleaned. Data Cleaning process helps to eliminate inconsistencies in the dataset. We remove things like special characters ,headings, newline characters, convert the text to lower case etc. We use UTF-8 encoding while reading the data from files to make sure consistency is maintained across all the documents.

Constructing Inverted Index

In this step we create an inverted index. An inverted index is a data structure used to search and efficiently retrieve information based on keywords. It works by mapping each unique word or term in a set of documents to the documents that contain it. Here are the steps to construct an inverted index.

Tokenization: Break each document into a sequence of words or terms. This process involves removing punctuation, converting all characters to lowercase, and removing stop words (common words like “and,“ “the,“ “a,“ etc. that do not provide meaningful information).

Create a dictionary: Create a dictionary that maps each unique word or term to an empty list. This dictionary will serve as the inverted index.

Posting: For each term in a document, add the document to the list of documents associated with that term in the dictionary. This process is known as posting.

Sort the dictionary: Sort the inverted index (dictionary) by the terms in alphabetical order.

Calculating TF-IDF values

Using the above formula we calculate the tf-idf value for each token in the query. We use the inverted index created to easily obtain the count of documents in which the token is present

Ranking using Cosine Similarity

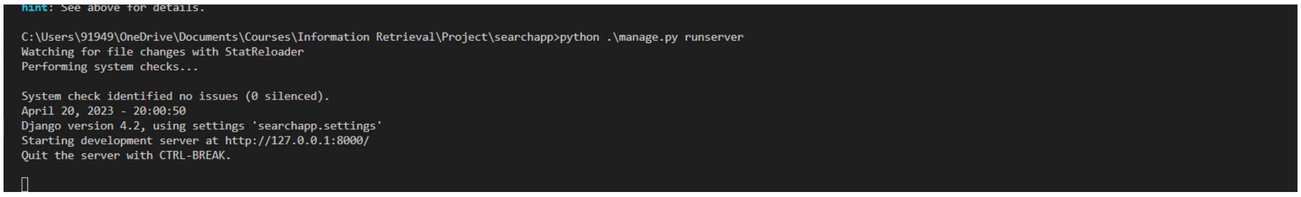
The simplest way to determine how similar two documents are is to use the matching score, which adds the tf_idf values of the tokens that are in the query for each document. For instance, we must determine whether a word is included in each document, and if it is, the tf_idf value is added to the matching score of that specific doc_id. The top k documents will be selected after sorting.

When we provide lengthy queries, Matching Score fails miserably because it cannot properly score the pages it provides as relevant. Cosine similarly measures the similarity in cosine space (the angle between the vectors) by marking all the documents as vectors of tf-idf tokens. Cosine similarity is the best way to determine relevance when the query length is modest but it may be strongly related to the document.

Analyzing the results

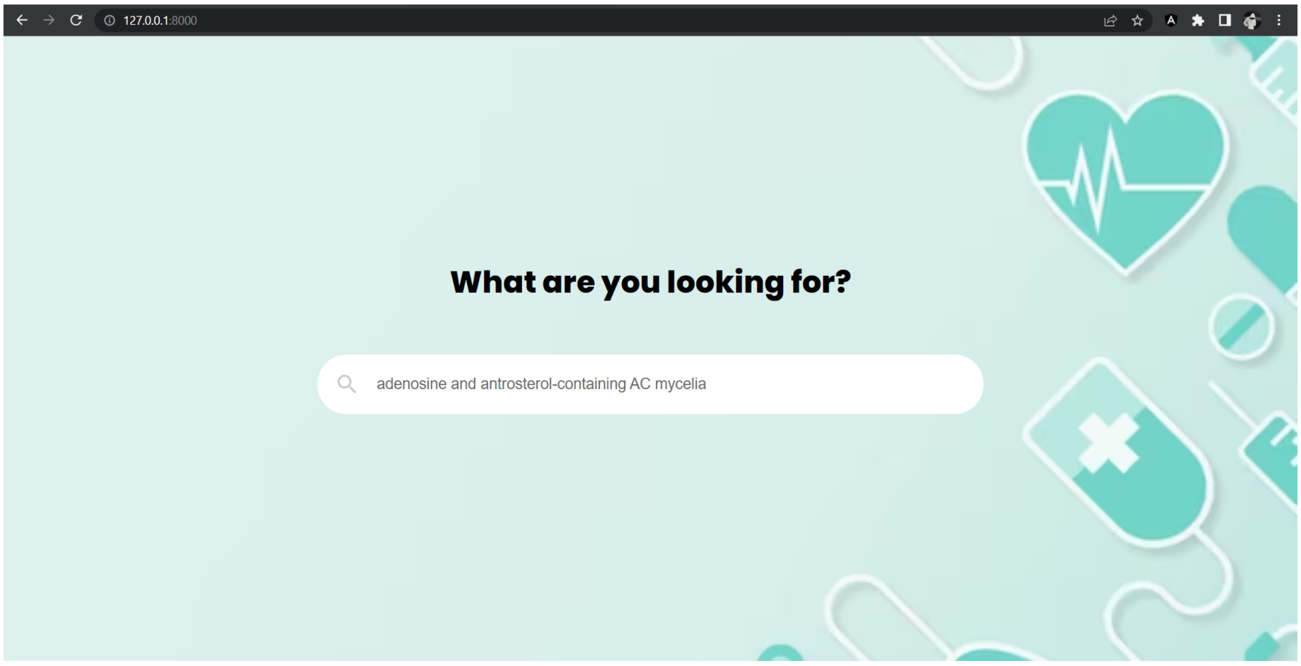
Here we view the resultant documents that are retrieved by the model. We are displaying the top 10 highly matching documents.

1. **Run the Django Server**



```
hint: See above for details.

C:\Users\91949\OneDrive\Documents\Courses\Information Retrieval\Project\searchapp>python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 20, 2023 - 20:00:50
Django version 4.2, using settings 'searchapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

2. **Enter the Search Query**

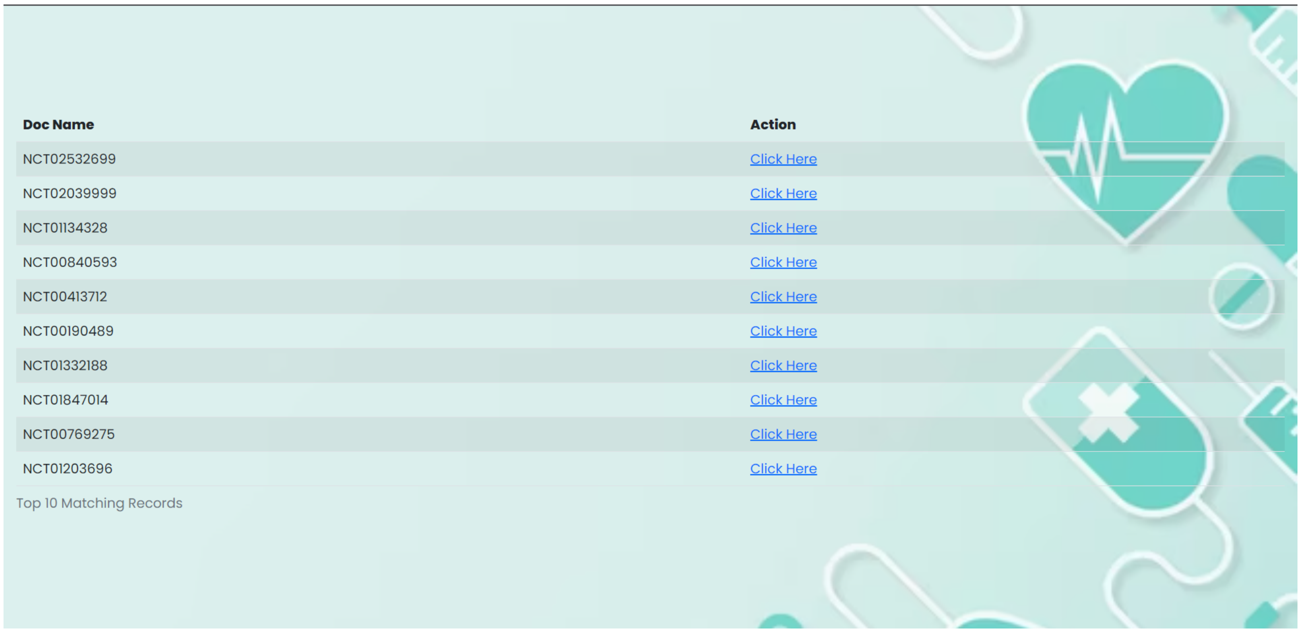


127.0.0.1:8000

What are you looking for?

adenosine and antrosterol-containing AC mycelia

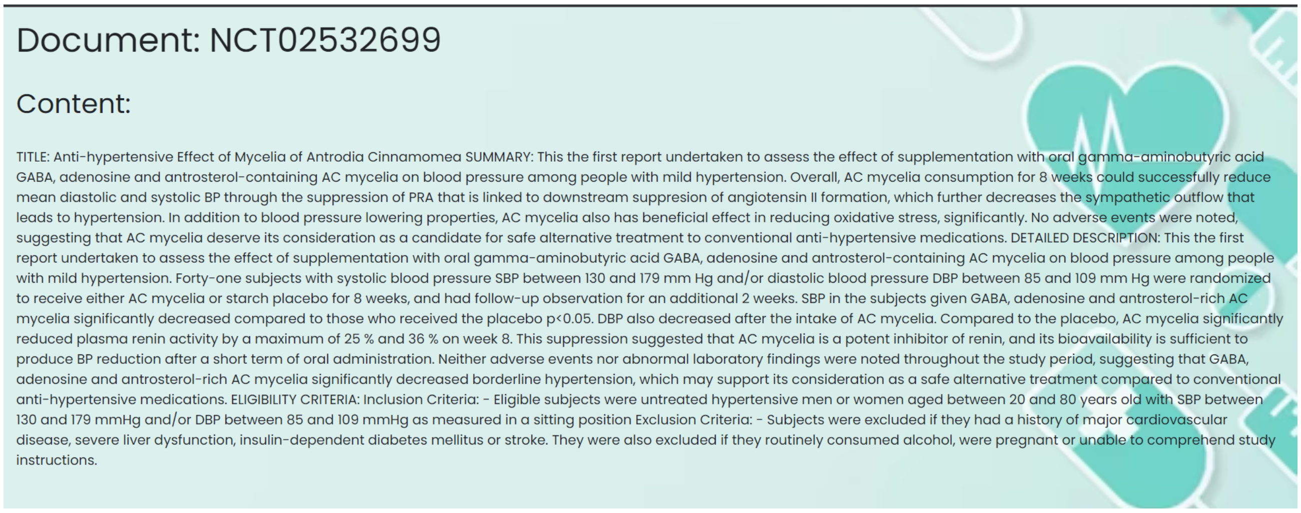
3. **Top 10 Documents**



Doc Name	Action
NCT02532699	Click Here
NCT02039999	Click Here
NCT01134328	Click Here
NCT00840593	Click Here
NCT00413712	Click Here
NCT00190489	Click Here
NCT01332188	Click Here
NCT01847014	Click Here
NCT00769275	Click Here
NCT01203696	Click Here

Top 10 Matching Records

4. **View the contents of Documents**

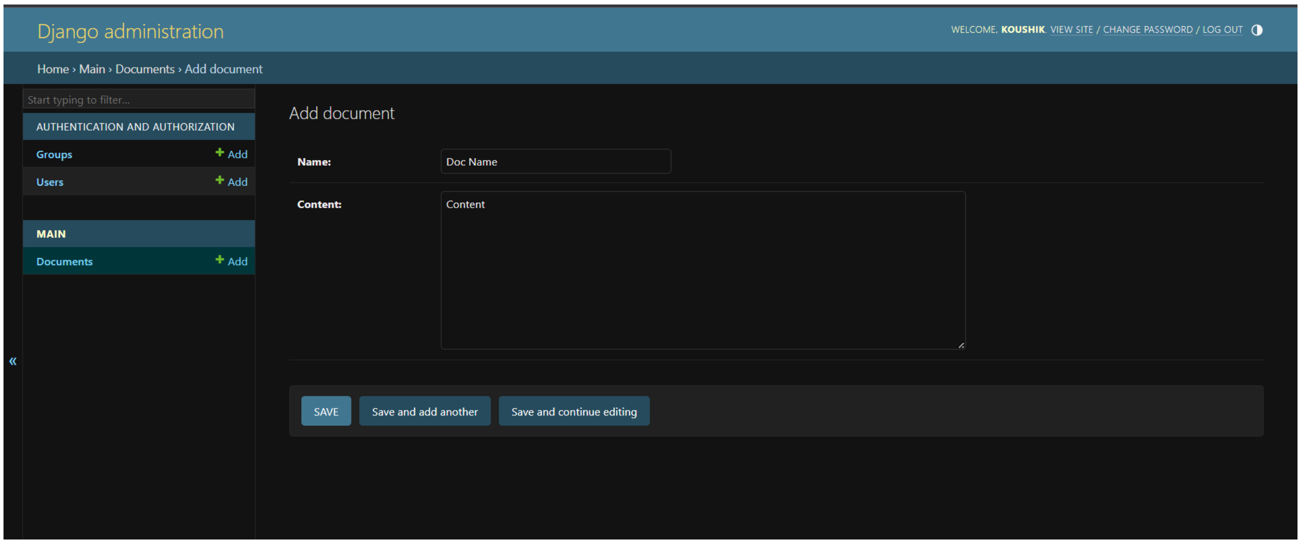


Document: NCT02532699

Content:

TITLE: Anti-hypertensive Effect of Mycelia of Antrodia Cinnamomea SUMMARY: This the first report undertaken to assess the effect of supplementation with oral gamma-aminobutyric acid GABA, adenosine and antrosterol-containing AC mycelia on blood pressure among people with mild hypertension. Overall, AC mycelia consumption for 8 weeks could successfully reduce mean diastolic and systolic BP through the suppression of PRA that is linked to downstream suppresion of angiotensin II formation, which further decreases the sympathetic outflow that leads to hypertension. In addition to blood pressure lowering properties, AC mycelia also has beneficial effect in reducing oxidative stress, significantly. No adverse events were noted, suggesting that AC mycelia deserve its consideration as a candidate for safe alternative treatment to conventional anti-hypertensive medications. DETAILED DESCRIPTION: This the first report undertaken to assess the effect of supplementation with oral gamma-aminobutyric acid GABA, adenosine and antrosterol-containing AC mycelia on blood pressure among people with mild hypertension. Forty-one subjects with systolic blood pressure SBP between 130 and 179 mm Hg and/or diastolic blood pressure DBP between 85 and 109 mm Hg were randomized to receive either AC mycelia or starch placebo for 8 weeks, and had follow-up observation for an additional 2 weeks. SBP in the subjects given GABA, adenosine and antrosterol-rich AC mycelia significantly decreased compared to those who received the placebo p<0.05. DBP also decreased after the intake of AC mycelia. Compared to the placebo, AC mycelia significantly reduced plasma renin activity by a maximum of 25 % and 36 % on week 8. This suppression suggested that AC mycelia is a potent inhibitor of renin, and its bioavailability is sufficient to produce BP reduction after a short term of oral administration. Neither adverse events nor abnormal laboratory findings were noted throughout the study period, suggesting that GABA, adenosine and antrosterol-rich AC mycelia significantly decreased borderline hypertension, which may support its consideration as a safe alternative treatment compared to conventional anti-hypertensive medications. ELIGIBILITY CRITERIA: Inclusion Criteria: - Eligible subjects were untreated hypertensive men or women aged between 20 and 80 years old with SBP between 130 and 179 mmHg and/or DBP between 85 and 109 mmHg as measured in a sitting position Exclusion Criteria: - Subjects were excluded if they had a history of major cardiovascular disease, severe liver dysfunction, insulin-dependent diabetes mellitus or stroke. They were also excluded if they routinely consumed alcohol, were pregnant or unable to comprehend study instructions.

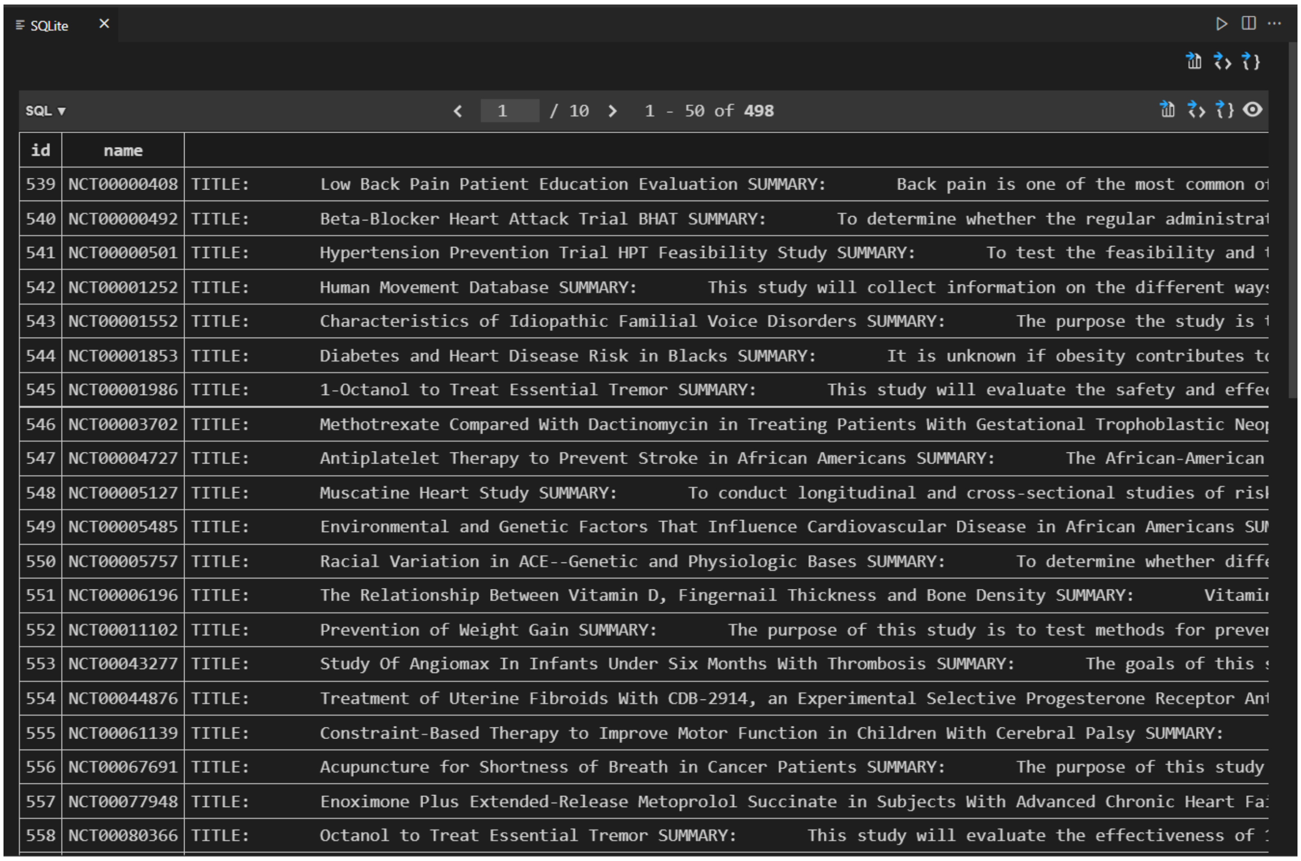
5. **Add new Documents**


Django administration
WELCOME, KOUSHIK. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Main › Documents › Add document
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
MAIN
Documents + Add
SAVE
Save and add another
Save and continue editing

Add document

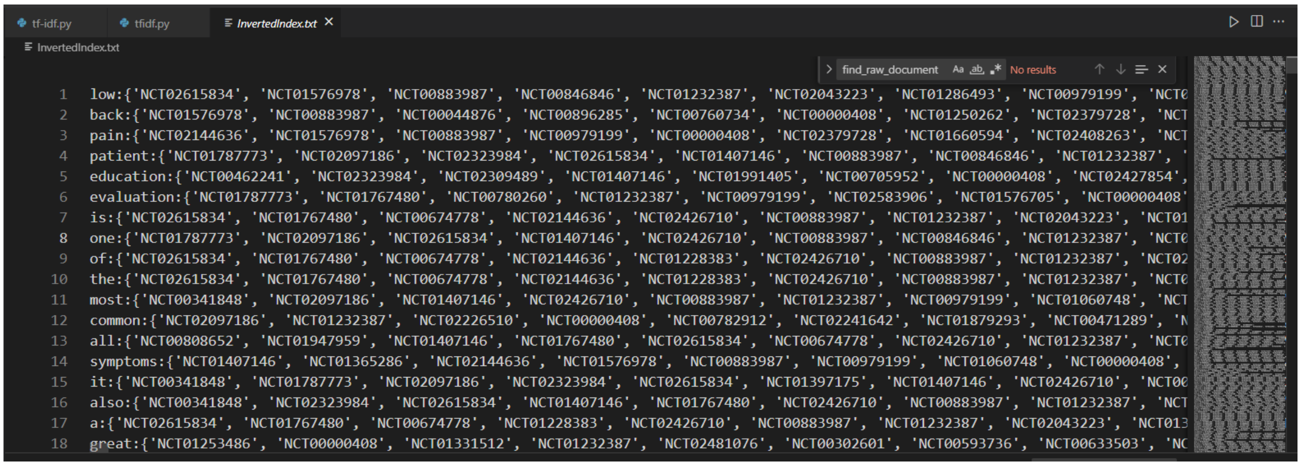
Name:	Doc Name
Content:	Content

6. **View Data in Database**



id	name	
539	NCT00000408	TITLE: Low Back Pain Patient Education Evaluation SUMMARY: Back pain is one of the most common o
540	NCT00000492	TITLE: Beta-Blocker Heart Attack Trial BHAT SUMMARY: To determine whether the regular administra
541	NCT00000501	TITLE: Hypertension Prevention Trial HPT Feasibility Study SUMMARY: To test the feasibility and
542	NCT00001252	TITLE: Human Movement Database SUMMARY: This study will collect information on the different way
543	NCT00001552	TITLE: Characteristics of Idiopathic Familial Voice Disorders SUMMARY: The purpose the study is
544	NCT00001853	TITLE: Diabetes and Heart Disease Risk in Blacks SUMMARY: It is unknown if obesity contributes t
545	NCT00001986	TITLE: 1-Octanol to Treat Essential Tremor SUMMARY: This study will evaluate the safety and effe
546	NCT00003702	TITLE: Methotrexate Compared With Dactinomycin in Treating Patients With Gestational Trophoblastic Neo
547	NCT00004727	TITLE: Antiplatelet Therapy to Prevent Stroke in African Americans SUMMARY: The African-American
548	NCT00005127	TITLE: Muscatine Heart Study SUMMARY: To conduct longitudinal and cross-sectional studies of ris
549	NCT00005485	TITLE: Environmental and Genetic Factors That Influence Cardiovascular Disease in African Americans SU
550	NCT00005757	TITLE: Racial Variation in ACE--Genetic and Physiologic Bases SUMMARY: To determine whether diff
551	NCT00006196	TITLE: The Relationship Between Vitamin D, Fingernail Thickness and Bone Density SUMMARY: Vitami
552	NCT00011102	TITLE: Prevention of Weight Gain SUMMARY: The purpose of this study is to test methods for preve
553	NCT00043277	TITLE: Study Of Angiomax In Infants Under Six Months With Thrombosis SUMMARY: The goals of this
554	NCT00044876	TITLE: Treatment of Uterine Fibroids With CDB-2914, an Experimental Selective Progesterone Receptor An
555	NCT00061139	TITLE: Constraint-Based Therapy to Improve Motor Function in Children With Cerebral Palsy SUMMARY:
556	NCT00067691	TITLE: Acupuncture for Shortness of Breath in Cancer Patients SUMMARY: The purpose of this study
557	NCT00077948	TITLE: Enoximone Plus Extended-Release Metoprolol Succinate in Subjects With Advanced Chronic Heart Fa
558	NCT00080366	TITLE: Octanol to Treat Essential Tremor SUMMARY: This study will evaluate the effectiveness of

7. **Inverted Index**

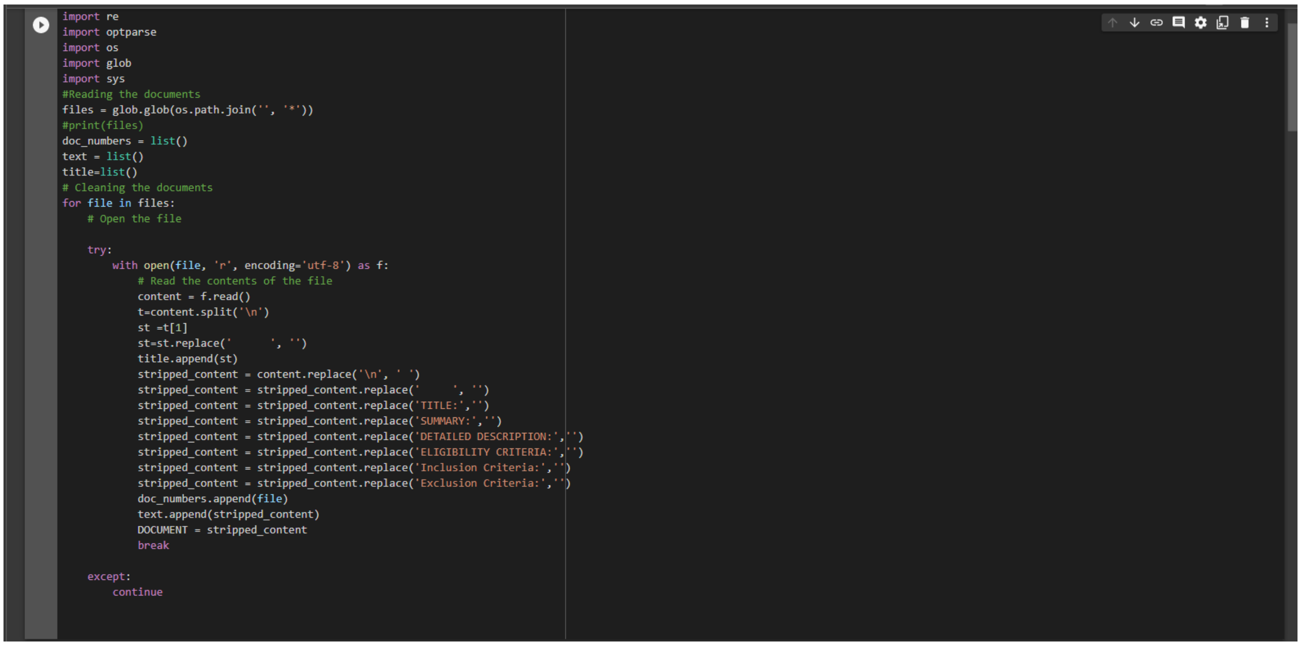


```
low:{'NCT02615834', 'NCT01576978', 'NCT00883987', 'NCT00846846', 'NCT01232387', 'NCT02043223', 'NCT01286493', 'NCT00979199', 'NCT0
back:{'NCT01576978', 'NCT00883987', 'NCT00044876', 'NCT00896285', 'NCT00760734', 'NCT00000408', 'NCT01250262', 'NCT02379728', 'NCT
pain:{'NCT02144636', 'NCT01576978', 'NCT00883987', 'NCT00979199', 'NCT00000408', 'NCT02379728', 'NCT01660594', 'NCT02408263', 'NCT
patient:{'NCT01787773', 'NCT02097186', 'NCT02323984', 'NCT02615834', 'NCT01407146', 'NCT00883987', 'NCT00846846', 'NCT01232387', '
education:{'NCT00462241', 'NCT02323984', 'NCT02309489', 'NCT01407146', 'NCT01991405', 'NCT00705952', 'NCT00000408', 'NCT02427854',
evaluation:{'NCT01787773', 'NCT01767480', 'NCT00780260', 'NCT01232387', 'NCT00979199', 'NCT02583906', 'NCT01576705', 'NCT00000408'
is:{'NCT02615834', 'NCT01767480', 'NCT00674778', 'NCT02144636', 'NCT02426710', 'NCT00883987', 'NCT01232387', 'NCT02043223', 'NCT01
one:{'NCT01787773', 'NCT02097186', 'NCT02615834', 'NCT01407146', 'NCT02426710', 'NCT00883987', 'NCT00846846', 'NCT01232387', 'NCT0
of:{'NCT02615834', 'NCT01767480', 'NCT00674778', 'NCT02144636', 'NCT01228383', 'NCT02426710', 'NCT00883987', 'NCT01232387', 'NCT02
the:{'NCT02615834', 'NCT01767480', 'NCT00674778', 'NCT02144636', 'NCT01228383', 'NCT02426710', 'NCT00883987', 'NCT01232387', 'NCT0
most:{'NCT00341848', 'NCT02097186', 'NCT01407146', 'NCT02426710', 'NCT00883987', 'NCT01232387', 'NCT00979199', 'NCT01060748', 'NCT
common:{'NCT02097186', 'NCT01232387', 'NCT02226510', 'NCT00000408', 'NCT00782912', 'NCT02241642', 'NCT01879293', 'NCT00471289', 'N
all:{'NCT00808652', 'NCT01947959', 'NCT01407146', 'NCT01767480', 'NCT02615834', 'NCT00674778', 'NCT02426710', 'NCT01232387', 'NCT0
symptoms:{'NCT01407146', 'NCT01365286', 'NCT02144636', 'NCT01576978', 'NCT00883987', 'NCT00979199', 'NCT01060748', 'NCT00000408',
it:{'NCT00341848', 'NCT01787773', 'NCT02097186', 'NCT02323984', 'NCT02615834', 'NCT01397175', 'NCT01407146', 'NCT02426710', 'NCT00
also:{'NCT00341848', 'NCT02323984', 'NCT02615834', 'NCT01407146', 'NCT01767480', 'NCT02426710', 'NCT00883987', 'NCT01232387', 'NCT
a:{'NCT02615834', 'NCT01767480', 'NCT00674778', 'NCT01228383', 'NCT02426710', 'NCT00883987', 'NCT01232387', 'NCT02043223', 'NCT013
great:{'NCT01253486', 'NCT00000408', 'NCT01331512', 'NCT01232387', 'NCT02481076', 'NCT00302601', 'NCT00593736', 'NCT00633503', 'NC
```

8. **Document Summarization**

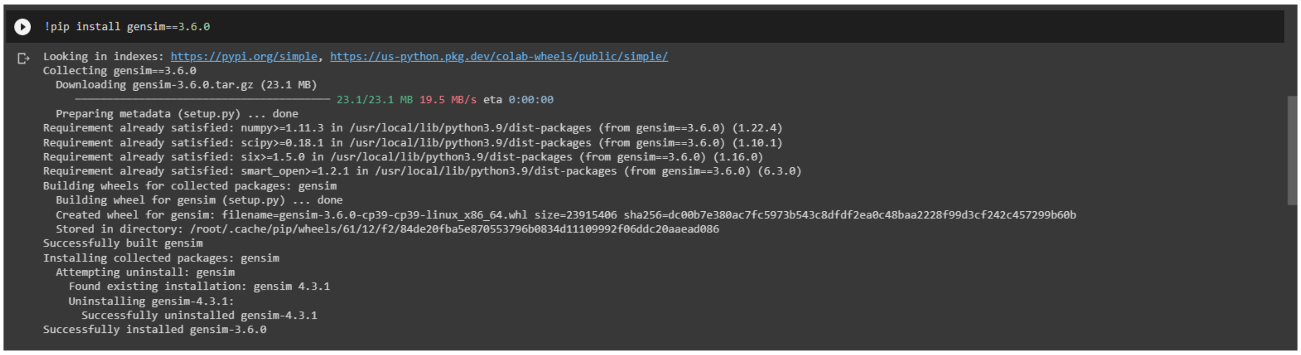
Here to summarize the large medical documents we are using the GENSIM summarizer. Gensim stands for "Generate Similar" is a popular open source natural language processing (NLP) library used for unsupervised topic modeling. It uses top academic models and modern statistical machine learning to perform various complex. Gensim is implemented in Python is designed to handle large text collections using data streaming as well as incremental online algorithms. This makes it different from those machine learning software packages that target only in-memory processing.

Cleaning the data set



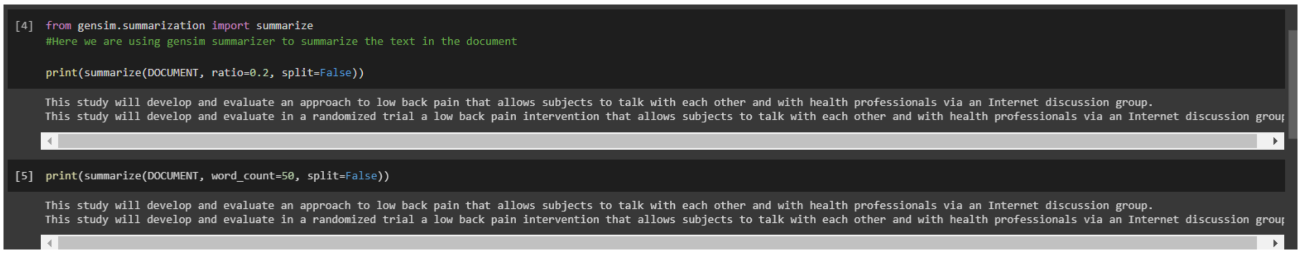
```python
import re
import optparse
import os
import glob
import sys
#Reading the documents
files = glob.glob(os.path.join('', '*'))
#print(files)
doc_numbers = list()
text = list()
title=list()
# Cleaning the documents
for file in files:
    # Open the file

    try:
        with open(file, 'r', encoding='utf-8') as f:
            # Read the contents of the file
            content = f.read()
            t=content.split('\n')
            st =t[1]
            st=st.replace('       ', '')
            title.append(st)
            stripped_content = content.replace('\n', ' ')
            stripped_content = stripped_content.replace('      ', '')
            stripped_content = stripped_content.replace('TITLE:','')
            stripped_content = stripped_content.replace('SUMMARY:','')
            stripped_content = stripped_content.replace('DETAILED DESCRIPTION:','')
            stripped_content = stripped_content.replace('ELIGIBILITY CRITERIA:','')
            stripped_content = stripped_content.replace('Inclusion Criteria:','')
            stripped_content = stripped_content.replace('Exclusion Criteria:','')
            doc_numbers.append(file)
            text.append(stripped_content)
            DOCUMENT = stripped_content
            break

    except:
        continue
```

Installing gensim using pip



```
!pip install gensim==3.6.0

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting gensim==3.6.0
  Downloading gensim-3.6.0.tar.gz (23.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.1/23.1 MB 19.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.9/dist-packages (from gensim==3.6.0) (1.22.4)
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.9/dist-packages (from gensim==3.6.0) (1.10.1)
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.9/dist-packages (from gensim==3.6.0) (1.16.0)
Requirement already satisfied: smart_open>=1.2.1 in /usr/local/lib/python3.9/dist-packages (from gensim==3.6.0) (6.3.0)
Building wheels for collected packages: gensim
  Building wheel for gensim (setup.py) ... done
  Created wheel for gensim: filename=gensim-3.6.0-cp39-cp39-linux_x86_64.whl size=23915406 sha256=dc00b7e380ac7fc5973b543c8dfdf2ea0c48baa2228f99d3cf242c457299b60b
  Stored in directory: /root/.cache/pip/wheels/61/12/f2/84de20fba5e870553796b0834d11109992f06ddc20aaead086
Successfully built gensim
Installing collected packages: gensim
  Attempting uninstall: gensim
    Found existing installation: gensim 4.3.1
    Uninstalling gensim-4.3.1:
      Successfully uninstalled gensim-4.3.1
Successfully installed gensim-3.6.0
```

Summarized documents



```
[4] from gensim.summarization import summarize
    #Here we are using gensim summarizer to summarize the text in the document

    print(summarize(DOCUMENT, ratio=0.2, split=False))

    This study will develop and evaluate an approach to low back pain that allows subjects to talk with each other and with health professionals via an Internet discussion group.
    This study will develop and evaluate in a randomized trial a low back pain intervention that allows subjects to talk with each other and with health professionals via an Internet discussion grou

[5] print(summarize(DOCUMENT, word_count=50, split=False))

    This study will develop and evaluate an approach to low back pain that allows subjects to talk with each other and with health professionals via an Internet discussion group.
    This study will develop and evaluate in a randomized trial a low back pain intervention that allows subjects to talk with each other and with health professionals via an Internet discussion grou
```

Project Milestones

The following are the milestones of this project.

1. Data Collection: Collect data from various sources such as medical literature databases, clinical trial datasets, and other relevant sources.

2. Data Preprocessing: The collected data to remove noise, normalize the data, and convert it into a structured format that can be analyzed and features are extracted.

3. Frontend Development : The UI part of the project is developed using HTML, CSS, Bootstrap enable end users to search for medical information and to show relevant information.

4. **Backend Development:** The supporting backend is developed for the system we used Django, Python, SQLite database.

5. Information Retrieval Model Development: Here we developed an information retrieval model using TF-IDF model that can search for relevant information in large datasets

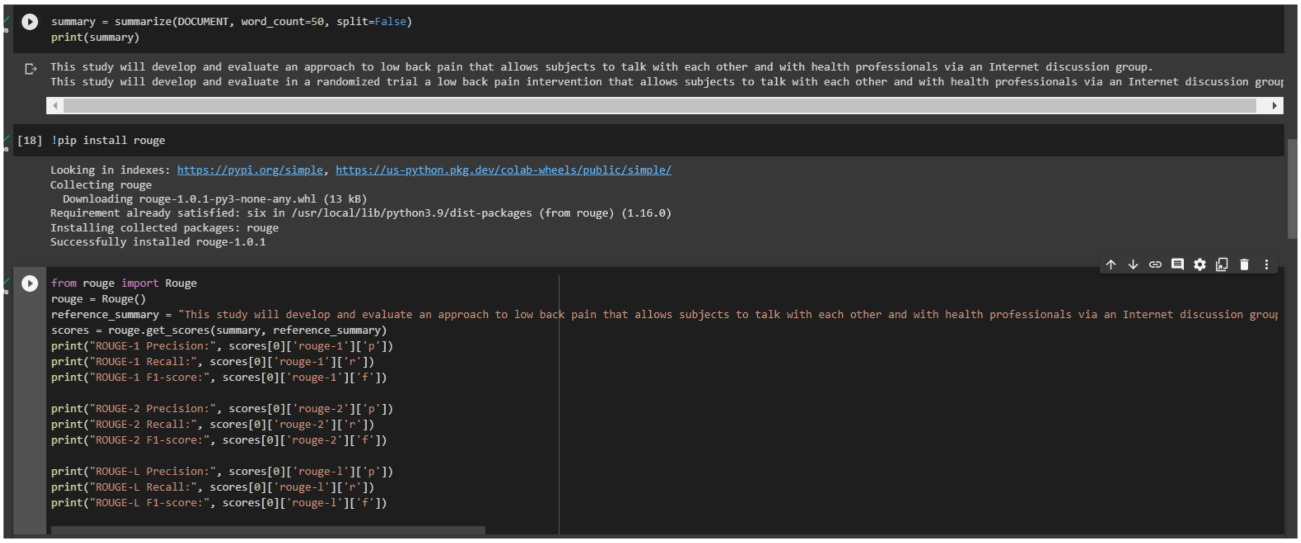
6. Text Summarization Model Development: Here we developed a text summarization model that can summarize relevant information from unstructured data, such as clinical notes or medical literature. We used the GENSIM model to summarize the medical information.

7. Evaluation: The eighth milestone is to evaluate the models and the models are evaluated using different evaluation metrics.

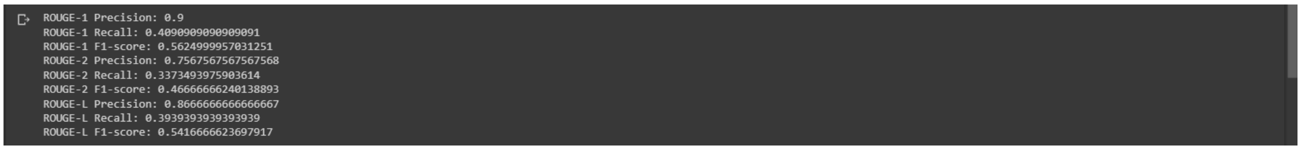
Evaluation Results

To evaluate the model we are using using cosine similarity we are ranking the order of the medical documents. We are using rouge metric to evaluate the summarizer and calculate the Precision, recall and F1 score. Recall-Oriented Understudy for Gisting Evaluation (ROUGE) is a widely used metric to evaluate the quality of automatic summarization systems. It measures the overlap between the generated summary and the reference summary in terms of n-gram overlap, sentence similarity, and other measures. Gensim provides a ROUGE implementation, which can be used to calculate ROUGE scores.

Evaluation of Summarizer

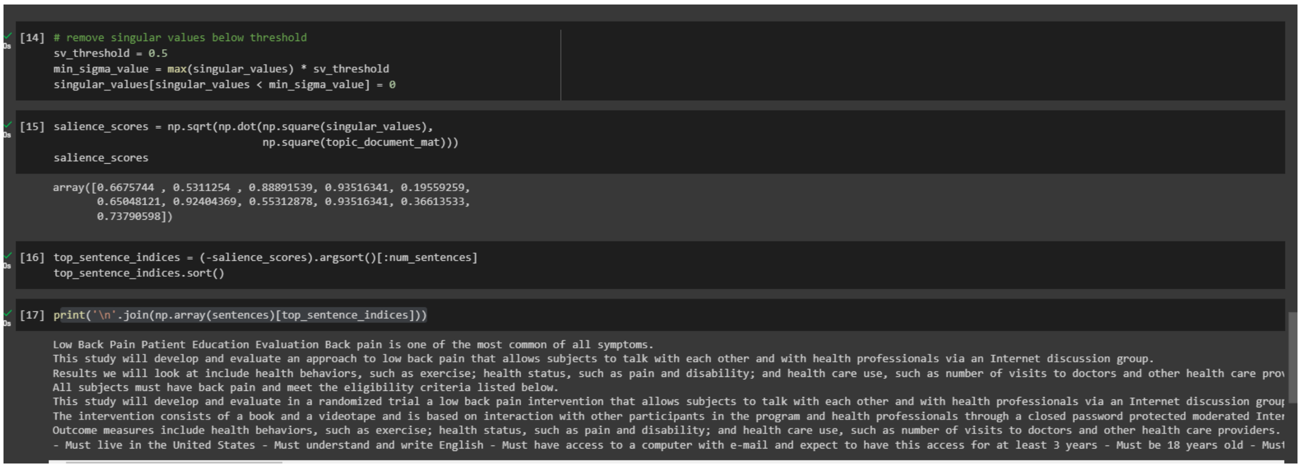


```
summary = summarize(DOCUMENT, word_count=50, split=False)
print(summary)

This study will develop and evaluate an approach to low back pain that allows subjects to talk with each other and with health professionals via an Internet discussion group.
This study will develop and evaluate in a randomized trial a low back pain intervention that allows subjects to talk with each other and with health professionals via an Internet discussion grou

[18] !pip install rouge

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting rouge
  Downloading rouge-1.0.1-py3-none-any.whl (13 kB)
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from rouge) (1.16.0)
Installing collected packages: rouge
Successfully installed rouge-1.0.1

from rouge import Rouge
rouge = Rouge()
reference_summary = "This study will develop and evaluate an approach to low back pain that allows subjects to talk with each other and with health professionals via an Internet discussion grou
scores = rouge.get_scores(summary, reference_summary)
print("ROUGE-1 Precision:", scores[0]['rouge-1']['p'])
print("ROUGE-1 Recall:", scores[0]['rouge-1']['r'])
print("ROUGE-1 F1-score:", scores[0]['rouge-1']['f'])

print("ROUGE-2 Precision:", scores[0]['rouge-2']['p'])
print("ROUGE-2 Recall:", scores[0]['rouge-2']['r'])
print("ROUGE-2 F1-score:", scores[0]['rouge-2']['f'])

print("ROUGE-L Precision:", scores[0]['rouge-l']['p'])
print("ROUGE-L Recall:", scores[0]['rouge-l']['r'])
print("ROUGE-L F1-score:", scores[0]['rouge-l']['f'])
```



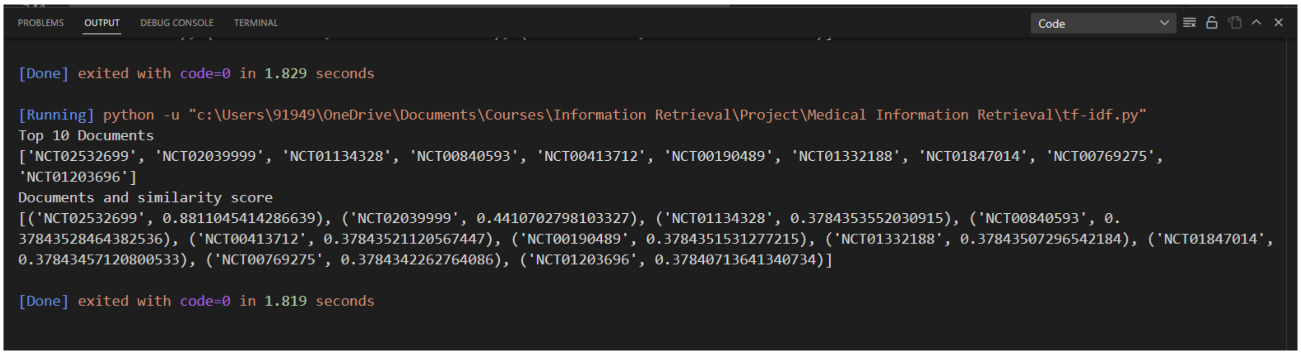
```
ROUGE-1 Precision: 0.9
ROUGE-1 Recall: 0.4090909090909091
ROUGE-1 F1-score: 0.5624999957031251
ROUGE-2 Precision: 0.7567567567567568
ROUGE-2 Recall: 0.3373493975903614
ROUGE-2 F1-score: 0.46666666240138893
ROUGE-L Precision: 0.8666666666666667
ROUGE-L Recall: 0.3939393939393939
ROUGE-L F1-score: 0.5416666623697917
```

Latent Semantic analysis



```
[14] # remove singular values below threshold
     sv_threshold = 0.5
     min_sigma_value = max(singular_values) * sv_threshold
     singular_values[singular_values < min_sigma_value] = 0

[15] salience_scores = np.sqrt(np.dot(np.square(singular_values),
                                     np.square(topic_document_mat)))
     salience_scores

     array([0.6675744 , 0.5311254 , 0.88891539, 0.93516341, 0.19559259,
            0.65048121, 0.92404369, 0.55312878, 0.93516341, 0.36613533,
            0.73790598])

[16] top_sentence_indices = (-salience_scores).argsort()[:num_sentences]
     top_sentence_indices.sort()

[17] print('\n'.join(np.array(sentences)[top_sentence_indices]))

     Low Back Pain Patient Education Evaluation Back pain is one of the most common of all symptoms.
     This study will develop and evaluate an approach to low back pain that allows subjects to talk with each other and with health professionals via an Internet discussion group.
     Results we will look at include health behaviors, such as exercise; health status, such as pain and disability; and health care use, such as number of visits to doctors and other health care prov
     All subjects must have back pain and meet the eligibility criteria listed below.
     This study will develop and evaluate in a randomized trial a low back pain intervention that allows subjects to talk with each other and with health professionals via an Internet discussion grou
     The intervention consists of a book and a videotape and is based on interaction with other participants in the program and health professionals through a closed password protected moderated Inter
     Outcome measures include health behaviors, such as exercise; health status, such as pain and disability; and health care use, such as number of visits to doctors and other health care providers.
     - Must live in the United States - Must understand and write English - Must have access to a computer with e-mail and expect to have this access for at least 3 years - Must be 18 years old - Mus
```

Results of TF-IDF Model



```
[Done] exited with code=0 in 1.829 seconds

[Running] python -u "c:\Users\91949\OneDrive\Documents\Courses\Information Retrieval\Project\Medical Information Retrieval\tf-idf.py"
Top 10 Documents
['NCT02532699', 'NCT02039999', 'NCT01134328', 'NCT00840593', 'NCT00413712', 'NCT00190489', 'NCT01332188', 'NCT01847014', 'NCT00769275',
'NCT01203696']
Documents and similarity score
[('NCT02532699', 0.8811045414286639), ('NCT02039999', 0.4410702798103327), ('NCT01134328', 0.3784353552030915), ('NCT00840593', 0.
37843528464382536), ('NCT00413712', 0.37843521120567447), ('NCT00190489', 0.3784351531277215), ('NCT01332188', 0.37843507296542184), ('NCT01847014',
0.37843457120800533), ('NCT00769275', 0.3784342262764086), ('NCT01203696', 0.37840713641340734)]

[Done] exited with code=0 in 1.819 seconds
```

References

He, Y., Jiang, M., Wang, J., & Huang, X. (2018). Medical text mining and its applicationsin academic medicine. Journal of Healthcare Engineering, 2018, 2710652. doi: 10.1155/2018/2710652

Sarker, A., & Gonzalez, G. (2015). Portable automatic text summarization for medical articles: A multi-document summarization approach. Journal of Biomedical Informatics, 53, 314-329. doi: 10.1016/j.jbi.2014.12.004

Roberts, K., Rodriguez, L., Shojaee, A., & Demner-Fushman, D. (2018). A comparative study of different summarization algorithms on a clinical corpus. Journal of Biomedical Informatics, 86, 76-86. doi: 10.1016/j.jbi.2018.07.009

Luo, Y., Uzuner, O., Szolovits, P., & Starren, J. (2014). Segment convolutional neural networks (Seg-CNNs) for classifying relations in clinical notes. Journal of the American Medical Informatics Association, 22(2), 253-265. doi: 10.1136/amiajnl-2013-002091

https://radimrehurek.com/gensim/intro.html#:~:text=Gensim%20is%20a%20free%20open,using%20unsupervised%20machine%20learning%20algorithms.

https://en.wikipedia.org/wiki/Tf%E2%80%93idf

https://scikit-learn.org/stable/modules/metrics.html#cosine-similarity

Project Code

Text Summarization

```
import math
from collections import Counter
import operator
import numpy as np
import re
import optparse
import os
import glob
import sys
#Reading the documents
files = glob.glob(os.path.join(’’, ’*’))
#print(files)
doc_numbers = list()
text = list()
title=list()
# Cleaning the documents
for file in files:
  # Open the file
  try:
    with open(file, ’r’, encoding=’utf-8’) as f:
      # Read the contents of the file
      content = f.read()
      t=content.split(’\n’)
      st =t[1]
      st=st.replace(’    ’, ’’)
      title.append(st)
      stripped_content = content.replace(’\n’, ’ ’)
      stripped_content = stripped_content.replace(’    ’, ’’)
```

```
            stripped_content = stripped_content.replace('TITLE:','')

            stripped_content = stripped_content.replace('SUMMARY:','')

            stripped_content = stripped_content.replace('DETAILED DESCRIPTION:','')

            stripped_content = stripped_content.replace('ELIGIBILITY CRITERIA:','')

            stripped_content = stripped_content.replace('Inclusion Criteria:','')

            stripped_content = stripped_content.replace('Exclusion Criteria:','')

            doc_numbers.append(file)

            text.append(stripped_content)

            DOCUMENT = stripped_content

            break

    except:

        continue

import re

DOCUMENT = re.sub(r'\n|\r', ' ', DOCUMENT)

DOCUMENT = re.sub(r' +', ' ', DOCUMENT)

DOCUMENT = DOCUMENT.strip()

!pip install gensim==3.6.0

from gensim.summarization import summarize

#Using the Gensim Summarizer

print(summarize(DOCUMENT, ratio=0.2, split=False))

summary = summarize(DOCUMENT, word_count=50, split=False)

print(summary)

!pip install rouge

from rouge import Rouge

rouge = Rouge()

reference_summary = "This study will develop and evaluate an approach to low back pain that allows subjects
to talk with each other and with health professionals via an Internet discussion group. Outcome measures
include health behaviors, health status, and health care use. Must have access to a computer with e-mail, be 18
years old, and have seen a doctor for back pain at least once in the past year. Back pain or sciatica can lead to
severe limitations, legal proceedings, bladder/bowel control issues, and numbness in the crotch."

scores = rouge.get_scores(summary, reference_summary)

print("ROUGE-1 Precision:", scores[0]['rouge-1']['p'])

print("ROUGE-1 Recall:", scores[0]['rouge-1']['r'])

print("ROUGE-1 F1-score:", scores[0]['rouge-1']['f'])

print("ROUGE-2 Precision:", scores[0]['rouge-2']['p'])

print("ROUGE-2 Recall:", scores[0]['rouge-2']['r'])
print(“ROUGE-2 F1-score:“, scores[0]['rouge-2']['f'])

print(“ROUGE-L Precision:“, scores[0]['rouge-l']['p'])

print(“ROUGE-L Recall:“, scores[0]['rouge-l']['r'])

print(“ROUGE-L F1-score:“, scores[0]['rouge-l']['f'])

from scipy.sparse.linalg import svds

def low_rank_svd(matrix, singular_count=2):

  u, s, vt = svds(matrix, k=singular_count)

  return u, s, vt

num_sentences = 8

num_topics = 3

u, s, vt = low_rank_svd(td_matrix, singular_count=num_topics)

print(u.shape, s.shape, vt.shape)

term_topic_mat, singular_values, topic_document_mat = u, s, vt

sv_threshold = 0.5

min_sigma_value = max(singular_values) * sv_threshold

singular_values[singular_values < min_sigma_value] = 0

salience_scores = np.sqrt(np.dot(np.square(singular_values),  np.square(topic_document_mat)))

salience_scores

top_sentence_indices = (-salience_scores).argsort()[:num_sentences]

top_sentence_indices.sort()

print('\n'.join(np.array(sentences)[top_sentence_indices]))

similarity_matrix = np.matmul(dt_matrix, dt_matrix.T)

print(similarity_matrix.shape)

np.round(similarity_matrix, 3)
```

TF-IDF

```
from bs4 import BeautifulSoup

from nltk.tokenize import RegexpTokenizer

import math

from collections import Counter

import operator

import numpy as np

import re

import optparse
import os

import glob

import sys

class IRModel:

  invertedIndex ={}

  def __init__(self, path2docs):

    self.docno, self.raw_documents = self.extract_text(path2docs)

    self.documents = self.preprocess(self.raw_documents)

    self.vocab = self.get_vocab(self.documents)

    self.N = len(self.documents)      # total number of documents

    self.invertedIndex = self.generate_inverted_index(self.docno,self.documents)

  def extract_text(self, path2docs):

    files = glob.glob(os.path.join(path2docs, '*'))

    #print(files)

    doc_numbers = list()

    text = list()

    for file in files:

      raw=[]

      try:

        with open(file, 'r', encoding='utf-8') as f:

          content = f.read()

          #Retrieving file name

          fname=file.split('\\')

          fname=fname[-1]

          doc_numbers.append(fname)

          stripped_content = content.replace('\n', ' ')

          stripped_content = stripped_content .replace('TITLE:','')

          stripped_content = stripped_content .replace('SUMMARY:','')

          stripped_content = stripped_content .replace('DETAILED DESCRIPTION:','')

          stripped_content = stripped_content .replace('ELIGIBILITY CRITERIA:','')

          stripped_content = stripped_content .replace('Inclusion Criteria:','')

          stripped_content = stripped_content .replace('Exclusion Criteria:','')

          raw.append(content)

          text.append(stripped_content)
        except:
       continue
    text.pop()
    raw.pop()
    doc_numbers.pop()
    return doc_numbers, text
def preprocess(self, text):
    tokenizer = RegexpTokenizer(r'\w+')
    preprocessed = list()
    for t in text:
        t = t.lower()
        preprocessed.append(tokenizer.tokenize(t))
    return preprocessed
def preprocess_str(self, sentence):
    tokenizer = RegexpTokenizer(r'\w+')
    sentence = tokenizer.tokenize(sentence.lower())
    return sentence
def get_vocab(self, text):
    vocab = list()
    for doc in text:
        vocab.extend(doc)
    set(vocab)
    return set(vocab)
def generate_inverted_index(self,docno,documents):
    for idx,doc in enumerate(documents):
        for token in doc:
            if token in self.invertedIndex:
                self.invertedIndex[token].add(docno[idx])
            else:
                self.invertedIndex[token] = {docno[idx]}
    #print(self.invertedIndex)
    with open('InvertedIndex.txt', 'w', encoding='utf-8') as f:
        for key,value in self.invertedIndex.items():
            f.write(key+':'+str(value)+'\n')
        return self.invertedIndex

    def idf(self, term):

        n_term = len(self.invertedIndex[term])

        if n_term == 0:

            return 0

        else:

            return math.log(self.N / n_term)

    def tf(self, term, doc):

        terms_in_doc = Counter(doc)

        max_term = max(terms_in_doc.values())

        return terms_in_doc[term] / max_term

    def get_vector(self, terms, document, idf_scores):

        vector = list()

        for term, idf in zip(terms, idf_scores):

            tf_idf = self.tf(term, document) * idf

            vector.append(tf_idf)

        return vector

    def similarity_scores(self, query):

        query = self.preprocess_str(query)

        idf_scores = [self.idf(term) for term in query]

        query_vec = self.get_vector(query, query, idf_scores)

        print(idf_scores)

        print(query_vec)

        # Get similarity scores for each document

        similarity_socres = dict()

        for doc, no in zip(self.documents, self.docno):

            doc_vec = self.get_vector(query, doc, idf_scores)

            # caculate the cosine similarity

            if np.dot(query_vec, doc_vec)!=0:

                cosine_sim = np.dot(query_vec, doc_vec) / (np.sqrt(np.sum(np.square(query_vec))) *
np.sqrt(np.sum(np.square(doc_vec))))

            else: cosine_sim = 0

            similarity_scores[no] = cosine_sim

        # Sorting
        similarity_scores = sorted(similarity_scores.items(), key=operator.itemgetter(1), reverse=True)

        return similarity_scores



def start(query):

    sys.stdout.reconfigure(encoding='utf-8')

    folder_path = 'C:\\Users\\91949\\OneDrive\\Documents\\Courses\\Information Retrieval\\Project\\Medical
Information Retrieval\\docs\\'

    path2docs = folder_path

    articles = IRModel(path2docs)

    queries = [query]

    # Rank total documents with the cosine similarity and tf-idf

    res=[]

    for q in queries:

        print(q)

        sim_scores = articles.similarity_scores(q)

        docno = [no for no, score in sim_scores]

        top_10_docs= docno[:10]

        res=top_10_docs

        print("Top 10 Documents")

        print(top_10_docs)

        print('Documents and similarity score')

        print(sim_scores[:10])

    return res
```

search_doc.html

```
<html>

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="author" content="colorlib.com">

    <link href="https://fonts.googleapis.com/css?family=Poppins:400,800" rel="stylesheet" />

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.min.js" integrity="sha384-
cuYeSxntonz0PPNlHhBs68uyIAVpIIOZZ5JqeqvYYIcEL727kskC66kF92t6Xl2V" crossorigin="anonymous"></script>
```

```
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4"
crossorigin="anonymous"></script>

</head>

  <body>

    <div class="s006">

        <table class="table table-striped" >

          <caption>Top 10 Matching Records</caption>

            <thead>

              <tr>

                <th>Doc Name</th>

                <th>Action</th>

              </tr>

            </thead>

            <tbody>

                {% for doc in docList %}

              <tr>

                <td>{{doc}}</td>

                <td><a href="doc/{{doc}}">Click Here</a></td>

              </tr>

              {% endfor %}

            </tbody>

          </table>

    </div>

  </body>

</html>
```

index.html

```
<html>

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="author" content="colorlib.com">

    <link href="https://fonts.googleapis.com/css?family=Poppins:400,800" rel="stylesheet" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" 
integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65" 
crossorigin="anonymous">

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.min.js" integrity="sha384-
cuYeSxntonz0PPNlHhBs68uyIAVpIIOZZ5JqeqvYYIcEL727kskC66kF92t6Xl2V" crossorigin="anonymous"></script>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" 
integrity="sha384-kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4" 
crossorigin="anonymous"></script>

</head>

 <body>

<div class="s006">

   <form method="POST" action="{% url 'search_doc' %}" name="searched">

    {% csrf_token %}

    <fieldset>

     <legend>What are you looking for?</legend>

     <div class="inner-form">

      <div class="input-field">

       <button class="btn-search" type="button">

        <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">

         <path d="M15.5 14h-.79l-.28-.27C15.41 12.59 16 11.11 16 9.5 16 5.91 13.09 3 9.5 3S3 5.91 3 9.5 5.91 16 9.5 
16c1.61 0 3.09-.59 4.23-1.57l.27.28v.79l5 4.99L20.49 19l-4.99-5zm-6 0C7.01 14 5 11.99 5 9.5S7.01 5 9.5 5 14 7.01 
14 9.5 11.99 14 9.5 14z"></path>

        </svg>

       </button>

       <input id="search" name="query" type="text" placeholder="Enter your search query" value="" />

      </div>

     </div>

    </fieldset>

   </form>

  </div>

 </body>

</html>
```

doc_details.html

```
<html>

 <head>

  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />
```

```
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />

   <meta name="author" content="colorlib.com">

   <link href="https://fonts.googleapis.com/css?family=Poppins:400,800" rel="stylesheet" />

   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">

   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.min.js" integrity="sha384-
cuYeSxntonz0PPNlHhBs68uyIAVpIIOZZ5JqeqvYYIcEL727kskC66kF92t6Xl2V" crossorigin="anonymous"></script>

   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4"
crossorigin="anonymous"></script>

   </head>

   <style>

     .s006 {

    min-height: 100vh;

    display: -ms-flexbox;

    -ms-flex-pack: center;

      justify-content: center;

    -ms-flex-align: center;

      align-items: center;

    font-family: 'Poppins', sans-serif;

    background: url("https://img.freepik.com/free-vector/clean-medical-background_53876-116875.jpg");

    background-size: cover;

    background-position: center center;

    padding: 15px;

   }

   </style>

<body>

   <div class="s006" >

     <h1>Document: {{doc.name}}</h1>

     <br/>

     <div>

     <h2>Content: </h2>

     </div>

     </br>

     <p>{{doc.content}}</p>
```

```
  </div>
 </body>
</html>
```

views.py

```
from django.shortcuts import render
from django.http import  HttpResponse
from . import tfidf
# Create your views here.
from .models import Document


def index(request):
  return render(request,’index.html’)


def doc_by_id(request,doc):


  doc1 = Document.objects.get(name=doc)
  if(doc1):
    return render(request,’doc_details.html’,{’doc’:doc1})
  else:
    return render(request,’doc_details.html’,{})


def search_doc(request):
  if(request.method==“POST“):
    query = request.POST.get(“query“, ““)
    li = tfidf.start(query)
    # res=’’
    # for doc in li:
    #    res+= doc
    return render(request,’search_doc.html’,{’query’:query,’docList’:li})
  else:
    return render(request,’search_doc.html’,{})
```

models.py

```
from django.db import models
# Create your models here.
```

```
class Document(models.Model):
   name = models.CharField(max_length=200)
   content =  models.TextField()
```

urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path(““,views.index,name=“index“),
    path(“doc/<str:doc>“,views.doc_by_id,name=’doc_by_id’),
    path(“search_doc“,views.search_doc,name=’search_doc’)
      ]
```