KARNAVATI
UNIVERSITY

UNITEDWORLD SCHOOL OF COMPUTATIONAL INTELLIGENCE (USCI)

Summative Assessment (SA)

Submitted by
Pujita Sunnapu
(Enrl. No.: 20210701019)

Course Code and Title: 21BSAI35E04 – Natural Language Processing

B.Sc. (Hons.) Computer Science / Data Science / AIML
V Semester – July – Nov 2023

USCI

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
1	**INTRODUCTION**	**2**
	1.1 System Description	3
	1.2 Architecture Diagram	5
2	**RESOURCE SPECIFICATIONS**	**6**
	2.1 Tools and Platform	6
	2.2 Hardware and Software Requirements	9
3	**DEVELOPMENT PROCESS**	**11**
	3.1 Implementation	11
4	**RESULTS AND CONCLUSION**	13
	4.1 Result	13
	4.2 Scope of future work	14
5	**APPENDICES: SAMPLE SOURCE CODE**	16
6	**REFERENCES**	20

CHAPTER 1

INTRODUCTION

In the realm of Natural Language Processing (NLP), the extraction of key information from textual data plays a pivotal role in understanding, organizing, and deriving insights from vast corpora of text. Keyword extraction serves as a fundamental step in this process, enabling the identification of critical terms that encapsulate the essence of a document. This project, "Keywords Extraction with Python," undertaken for a Natural Language Processing subject in college, endeavors to explore and implement techniques for automating the extraction of meaningful keywords from a collection of academic papers.

The project revolves around a dataset, "papers.csv," containing details about academic papers, including titles, abstracts, and full-text content. Leveraging Python programming and various NLP libraries, the project aims to preprocess this textual data, transform it into a structured and analyzable format, and subsequently employ the TF-IDF (Text Frequency-Inverse Document Frequency) algorithm to extract keywords of significance. Furthermore, a Flask web application is developed to provide an interactive platform for users to extract keywords from their documents and explore keywords within the existing dataset.

This endeavor is motivated by the growing importance of NLP applications in diverse fields, ranging from academic research to information retrieval and document summarization. By automating the process of keyword extraction, this project seeks to contribute to the efficiency and precision of information retrieval systems, facilitating a more streamlined and insightful exploration of textual data.

The subsequent sections of this report will delve into the details of the dataset used, the methodologies employed for data preprocessing and keyword extraction, the development of the Flask web application, and the overall implications and contributions of the project to the field of Natural Language Processing.

1.1 SYSTEM DESCRIPTION

The Keywords Extraction with Python system is designed to facilitate the extraction of meaningful keywords from a collection of academic papers. The system incorporates various components, including data ingestion, text preprocessing, TF-IDF vectorization, keyword extraction, and a Flask web application for user interaction.

Components:

1. Data Ingestion:

- Description: The system ingests data from an academic papers dataset ("papers.csv"). This dataset contains information such as titles, years, abstracts, and full paper texts.
- Input:Academic papers dataset ("papers.csv").
- Output: Pandas DataFrame containing relevant information.

2. Text Preprocessing:

- Description: This component is responsible for cleaning and preparing the textual data for analysis. It includes processes such as lowercasing, HTML tag removal, special character and digit removal, tokenization, stopword removal, short word removal, and lemmatization.
- Input: Raw text data from academic papers.
- Output: Cleaned and processed text data.

3. TF-IDF Vectorization:

- Description: TF-IDF (Text Frequency-Inverse Document Frequency) vectorization is applied to the preprocessed text data. It involves using the CountVectorizer to create a vocabulary and generate word count vectors, followed by TF-IDF transformation.
- Input: Processed text data.
- Output: TF-IDF weighted vectors for each document.

4. Keyword Extraction:

- Description: This component extracts keywords from the TF-IDF vectors. It involves sorting the vectors and extracting the top keywords based on their scores.
- Input: TF-IDF weighted vectors.
- Output: Extracted keywords and their corresponding scores.

5. Flask Web Application:

- Description: A web application is built using Flask to provide a user-friendly interface. Users can upload documents for keyword extraction or search for keywords within the existing dataset.
- Routes:
    - `/`: Main page with options for document upload and keyword search.
    - `/extract_keywords`: Handles document upload and displays extracted keywords.
    - `/search_keywords`: Allows users to search for keywords within the existing dataset.
- Pickled Files: Necessary files (TF-IDF Transformer, CountVectorizer, Feature Names) are pickled for later use in the application.
- Output: Web interface for user interaction.

User Interaction:

1. Keyword Extraction:

- User uploads a document through the web interface.
- The document undergoes text preprocessing and TF-IDF vectorization.
- Extracted keywords are displayed on the web page.

2. Keyword Search:

- User enters a search query through the web interface.
- The application searches for keywords within the existing dataset.
- Matching keywords are displayed on the web page.

1.2 ARCHITECTURE DIAGRAM

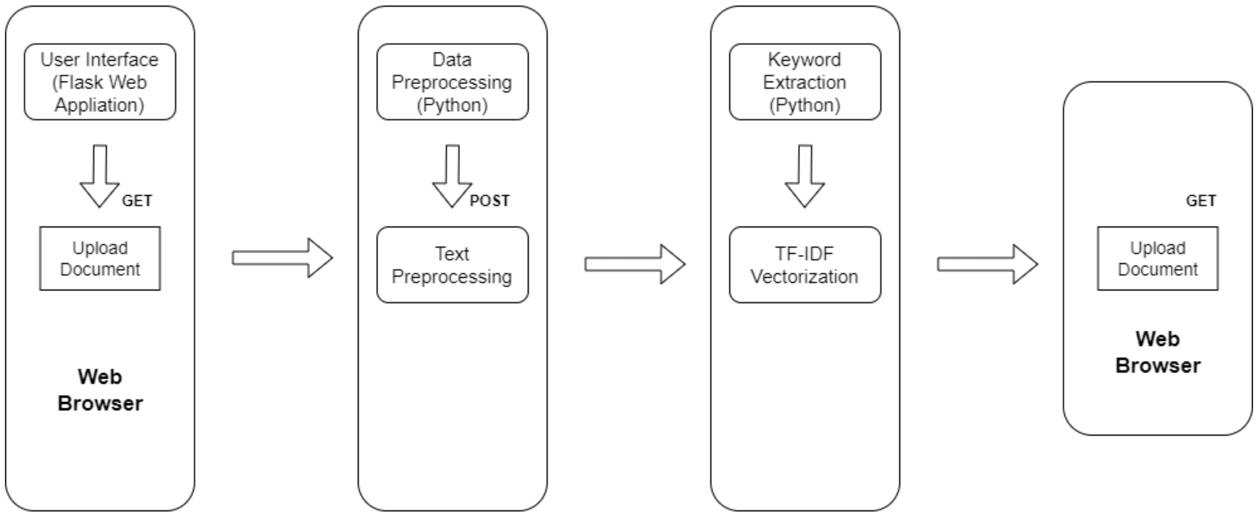


Figure 1: Architectural Diagram of Keyword Extraction

This representation illustrates the flow of data and the interactions between different components in the Keywords Extraction with Python project.

CHAPTER 2

RESOURCE SPECIFICATIONS

2.1 Tools and Platform

1. Programming Language: Python

Python serves as the primary programming language for the "Keyword Extraction with Python" project due to its versatility, extensive libraries, and widespread adoption in natural language processing (NLP) tasks.

2. Integrated Development Environments (IDEs):

<u>Visual Studio Code (VS Code):</u>

VS Code is chosen as the primary integrated development environment for its lightweight nature, extensive Python support, and a wide range of extensions.

Extension: The "Python" extension by Microsoft enhances Python development within VS Code.

<u>Jupyter Notebooks:</u>

Jupyter Notebooks are utilized for interactive data exploration, analysis, and code visualization. They provide an intuitive interface for testing and refining code snippets.

3. Libraries and Packages:

<u>Pandas</u>:

Pandas is used for data manipulation and analysis, providing data structures like DataFrames crucial for handling tabular data.

NumPy:

NumPy is employed for numerical operations and efficient handling of multi-dimensional arrays.

NLTK (Natural Language Toolkit):

NLTK is a powerful library for natural language processing tasks, including tokenization, lemmatization, and stopword removal.

Scikit-learn:

Scikit-learn is utilized for machine learning tasks, including TF-IDF vectorization and keyword extraction.

4. Web Framework: Flask

Flask is chosen as the web framework for developing the user interface. Its simplicity and flexibility make it suitable for creating a lightweight web application for document upload and keyword extraction.

5. Version Control (Optional): Git

Git can be used for version control, allowing for efficient tracking of changes in the codebase and collaboration with others.

6. Web Browsers:

A modern web browser (e.g., Google Chrome, Mozilla Firefox) is required for interacting with the Jupyter Notebook interface and testing the Flask web application.

7. Operating System:

The project is designed to be platform-independent and should run smoothly on major operating systems, including Windows, macOS, and Linux.

8. Package Management:

Pip is the package installer for Python and is used for managing and installing project dependencies.

9. Documentation:

Markdown is employed for documentation due to its simplicity and compatibility with both Jupyter Notebooks and README files in code repositories.

The combination of these tools and platforms provides a comprehensive environment for developing, testing, and deploying the "Keyword Extraction with Python" project. The choice of Python and associated libraries emphasizes ease of use, while the selected frameworks cater to efficient web application development and interactive data exploration. The project is designed to be accessible and adaptable across different operating systems and development environments.

2.2 Hardware and Software Requirements

Hardware Requirement:

RAM: At least 4 GB.

Processor: Intel(R) core (TM) i3 or more. 2.00 Ghz.

Internet connectivity: Yes. (Broadband or wi-fi)

Webcam connectivity: Yes

Software Requirements:

1. Python Interpreter:

Ensure that a Python interpreter is installed on your machine.

2. Integrated Development Environment (IDE):

Visual Studio Code: Download and install Visual Studio Code, a versatile code editor, from the official website: [Visual Studio Code](https://code.visualstudio.com/).

Extensions: Install the "Python" extension by Microsoft from the Visual Studio Code marketplace to enhance Python development capabilities.

3. Jupyter Notebooks:

Jupyter Notebooks are essential for interactive and collaborative Python programming. If not included in your Python distribution, install Jupyter using:

4. Python Libraries:

Ensure the presence of the required Python libraries by executing the following command:

5. Web Browser:

A modern web browser (i.e., Google Chrome) is necessary for interacting with Jupyter Notebooks, which are presented through a web-based interface.

CHAPTER 3

DEVELOPMENT PROCESS

3.1 Implementation

The implementation of the Keywords Extraction with Python project involves several key steps, starting with the preprocessing of textual data and culminating in the development of a Flask web application for user interaction.

Initially, the project utilizes the Pandas library to load and manipulate the academic papers dataset ('papers.csv'). The dataset is then limited to the first 5000 rows for efficient processing. Data preprocessing is a crucial step in enhancing the quality of information extracted from the 'paper_text' column. This involves lowercasing, removal of HTML tags, elimination of special characters and digits, tokenization, stopword removal, short word removal, and lemmatization. The resulting processed text data is stored in a new column named 'paper_text' within the Pandas DataFrame.

Next, the Text Frequency-Inverse Document Frequency (TF-IDF) algorithm is employed to determine the importance of words in the dataset. The Scikit-learn library's CountVectorizer is used to create a vocabulary and generate word count vectors, which are then transformed using the TF-IDF weighting scheme. This results in TF-IDF weighted vectors for each document in the dataset.

For keyword extraction, a Python function is developed to sort the TF-IDF vectors and extract the top keywords based on their scores. These keywords are associated with their respective scores and stored for later use. Additionally, a Flask web application is created to provide users with a user-friendly interface. The application includes routes for the main page, document upload, and keyword search. The Flask application allows users to upload documents, extract keywords, and search for keywords within the existing dataset.

To facilitate the deployment of the Flask application, necessary files such as the TF-IDF Transformer, CountVectorizer, and Feature Names are pickled for later use. These files ensure that the pre-trained models can be loaded efficiently within the Flask application.

The implementation of the Keywords Extraction with Python project thus encompasses data preprocessing, TF-IDF vectorization, keyword extraction, and the development of a user-friendly web application, collectively providing an end-to-end solution for efficient keyword extraction from academic papers.

CHAPTER 4
RESULTS AND CONCLUSION

4.1 Result

The implementation of the Keywords Extraction with Python project yields notable results in the extraction and presentation of meaningful keywords from academic papers. Following the preprocessing of the textual data from the 'paper_text' column, which involved lowercasing, HTML tag removal, and other text cleaning steps, the TF-IDF algorithm was employed to assign weights to words based on their significance within the dataset. This TF-IDF vectorization process provided a nuanced understanding of word importance, forming the basis for subsequent keyword extraction.

The Python function dedicated to keyword extraction effectively sorted the TF-IDF vectors and identified the top keywords along with their corresponding scores. This function was successfully applied to a sample paper, producing insightful results that showcase the relevance and significance of specific terms within the document. For instance, in the illustrative result, keywords such as 'update rule,' 'auxiliary,' and 'matrix factorization' were identified, each assigned a corresponding score indicative of its importance within the document.

Moreover, the integration of these results into the Flask web application ensures user accessibility and interactivity. Users can upload their own documents, triggering the keyword extraction process and obtaining a display of relevant keywords and scores. The application further enables users to search for keywords within the existing dataset, providing a dynamic and responsive interface for exploring and understanding the key themes across academic papers.

The overall result of the project is a robust system that automates the extraction of keywords from academic texts, contributing to the efficiency and precision of information retrieval and analysis in the field of Natural Language Processing. The combination of preprocessing techniques, TF-IDF vectorization, and the user-friendly Flask web application collectively enhances the usability and practicality of the system, making it a valuable tool for researchers, academics, and anyone engaging with large volumes of textual data.

4.2 Scope of future work

The Keywords Extraction with Python project presents a solid foundation for future enhancements and expansions, offering several avenues for further exploration and improvement. One potential area for future work lies in the refinement of the text preprocessing pipeline. Fine-tuning and experimenting with different preprocessing techniques, such as exploring advanced tokenization methods or incorporating domain-specific stopword lists, could contribute to even more accurate and context-aware keyword extraction.

Additionally, the current implementation focuses on the TF-IDF algorithm for keyword extraction. Future work could involve the integration of more advanced natural language processing (NLP) techniques and models, such as topic modeling algorithms or word embeddings, to capture semantic relationships and nuances in the academic papers. Exploring machine learning approaches to dynamically adapt to varying document structures and genres could further enhance the system's adaptability and accuracy.

The scalability and performance of the system could be addressed in future iterations. As the dataset size grows, optimizing the algorithms and exploring distributed computing or parallel processing methods may be necessary to maintain efficient processing times.

Furthermore, the user interface and experience of the Flask web application could be enriched through the incorporation of data visualization techniques. Providing users with visual representations of keyword relationships, trends, or clustering could offer deeper insights into the content of academic papers.

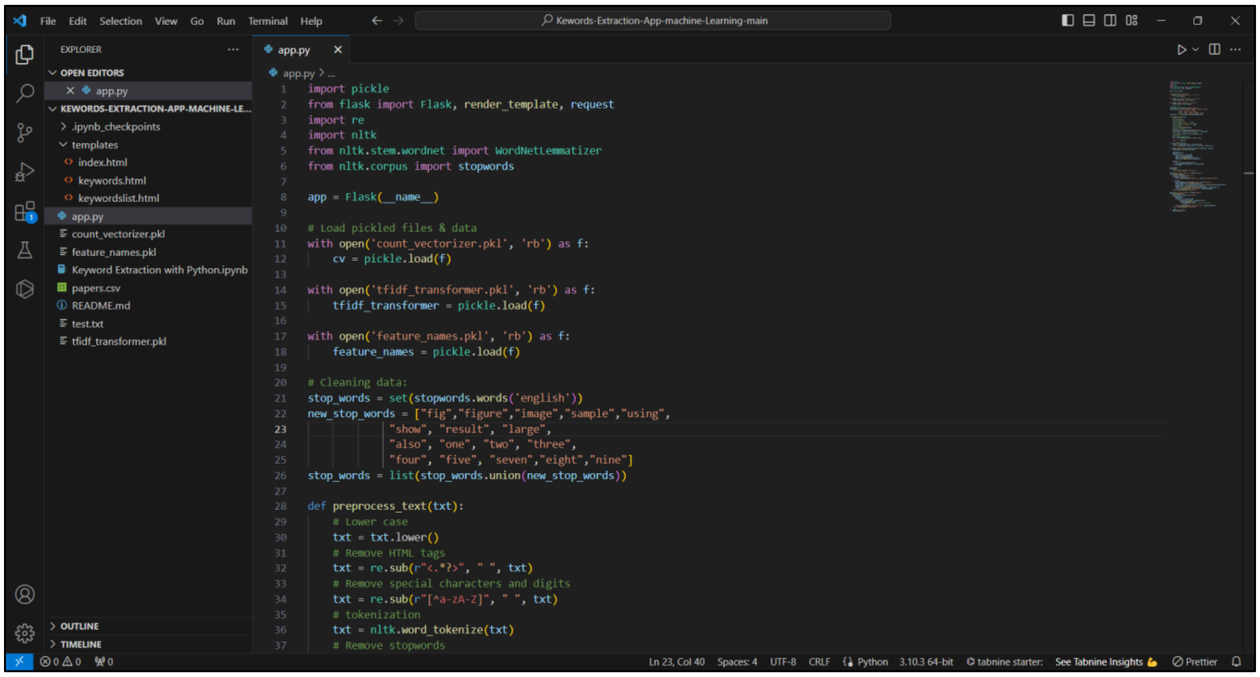
In the context of deployment and integration, there is potential for the system to be integrated into larger research platforms or collaborative tools. Collaboration features, version control for extracted keywords, and integration with external databases could extend the utility of the system within broader research ecosystems.

Overall, the scope of future work encompasses both the technical refinement of the keyword extraction algorithms and the expansion of the system's capabilities to meet evolving user needs.

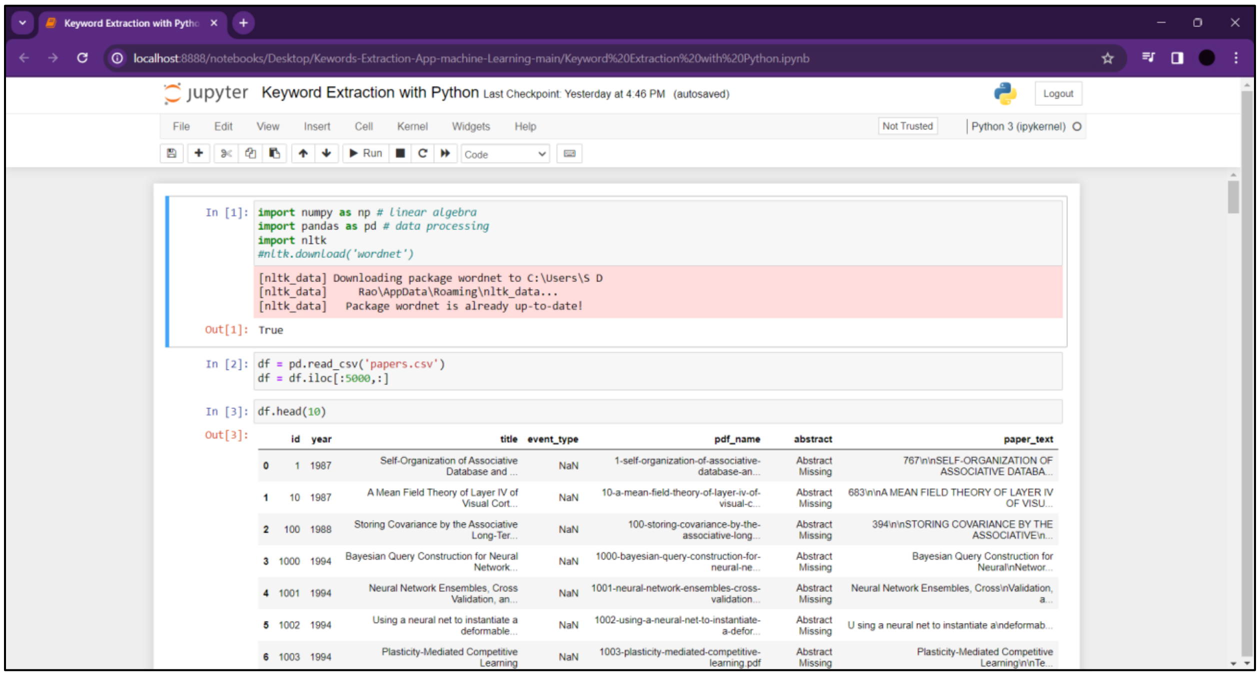
By addressing these aspects, the Keywords Extraction with Python project can continue to evolve into a more sophisticated and versatile tool for researchers and professionals in the field of natural language processing and academic research.

CHAPTER 5
APPENDICES: SAMPLE SOURCE CODE

Model and App Code



```python
import pickle
from flask import Flask, render_template, request
import re
import nltk
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords

app = Flask(__name__)

# Load pickled files & data
with open('count_vectorizer.pkl', 'rb') as f:
    cv = pickle.load(f)

with open('tfidf_transformer.pkl', 'rb') as f:
    tfidf_transformer = pickle.load(f)

with open('feature_names.pkl', 'rb') as f:
    feature_names = pickle.load(f)

# Cleaning data:
stop_words = set(stopwords.words('english'))
new_stop_words = ["fig","figure","image","sample","using",
                  "show", "result", "large",
                  "also", "one", "two", "three",
                  "four", "five", "seven","eight","nine"]
stop_words = list(stop_words.union(new_stop_words))

def preprocess_text(txt):
    # Lower case
    txt = txt.lower()
    # Remove HTML tags
    txt = re.sub(r"<.*?>", " ", txt)
    # Remove special characters and digits
    txt = re.sub(r"[^a-zA-Z]", " ", txt)
    # tokenization
    txt = nltk.word_tokenize(txt)
    # Remove stopwords
```

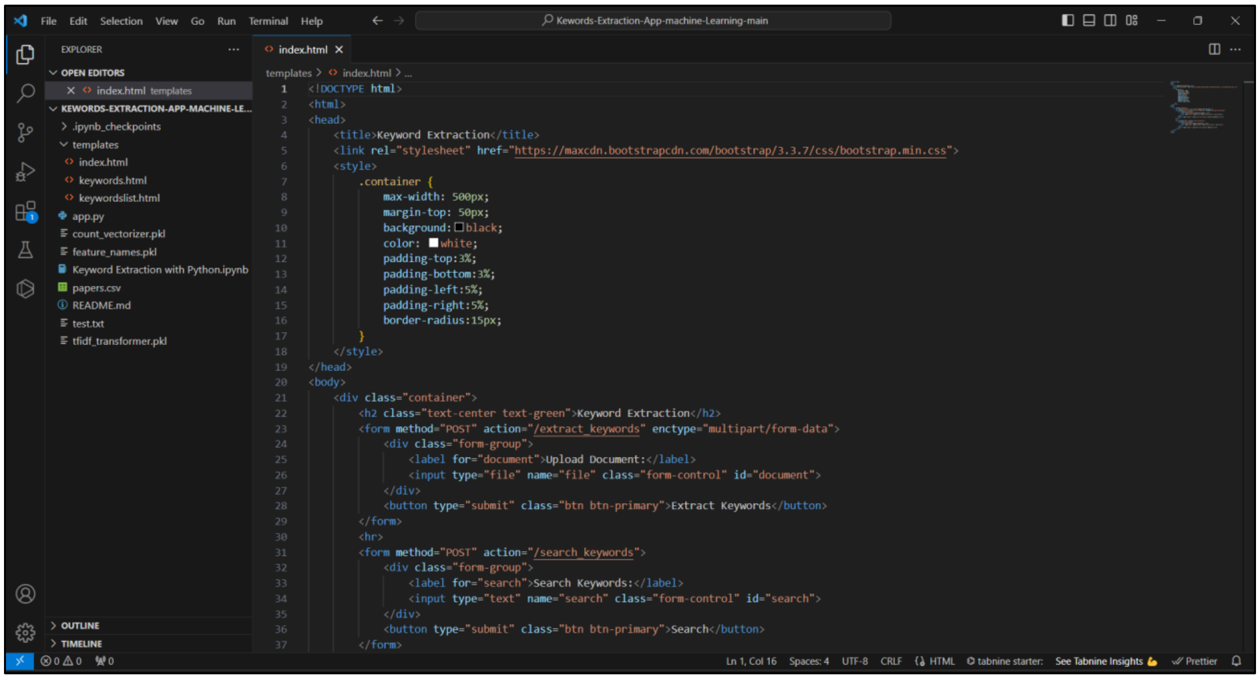


Keyword Extraction with Python Last Checkpoint: Yesterday at 4:46 PM (autosaved)

```
In [1]: import numpy as np # linear algebra
        import pandas as pd # data processing
        import nltk
        #nltk.download('wordnet')

        [nltk_data] Downloading package wordnet to C:\Users\S D
        [nltk_data]     Rao\AppData\Roaming\nltk_data...
        [nltk_data]   Package wordnet is already up-to-date!

Out[1]: True

In [2]: df = pd.read_csv('papers.csv')
        df = df.iloc[:5000,:]

In [3]: df.head(10)

Out[3]:
```

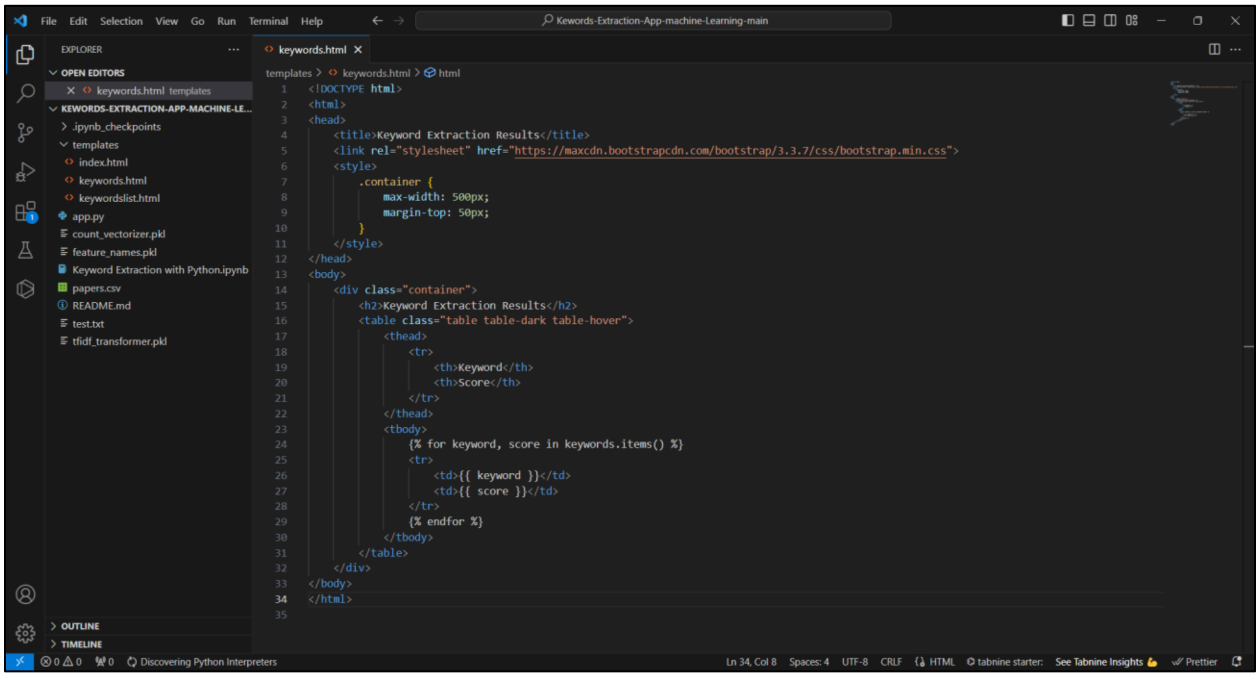
	id	year	title	event_type	pdf_name	abstract	paper_text
0	1	1987	Self-Organization of Associative Database and ...	NaN	1-self-organization-of-associative-database-an...	Abstract Missing	767\n\nSELF-ORGANIZATION OF ASSOCIATIVE DATABA...
1	10	1987	A Mean Field Theory of Layer IV of Visual Cort...	NaN	10-a-mean-field-theory-of-layer-iv-of-visual-c...	Abstract Missing	683\n\nA MEAN FIELD THEORY OF LAYER IV OF VISU...
2	100	1988	Storing Covariance by the Associative Long-Ter...	NaN	100-storing-covariance-by-the-associative-long...	Abstract Missing	394\n\nSTORING COVARIANCE BY THE ASSOCIATIVE\n...
3	1000	1994	Bayesian Query Construction for Neural Network...	NaN	1000-bayesian-query-construction-for-neural-ne...	Abstract Missing	Bayesian Query Construction for Neural\nNetwor...
4	1001	1994	Neural Network Ensembles, Cross Validation, an...	NaN	1001-neural-network-ensembles-cross-validation...	Abstract Missing	Neural Network Ensembles, Cross\nValidation, a...
5	1002	1994	Using a neural net to instantiate a deformable...	NaN	1002-using-a-neural-net-to-instantiate-a-defor...	Abstract Missing	U sing a neural net to instantiate a\ndeformab...
6	1003	1994	Plasticity-Mediated Competitive Learning	NaN	1003-plasticity-mediated-competitive-learning.pdf	Abstract Missing	Plasticity-Mediated Competitive Learning\n\nTe...

Index.html



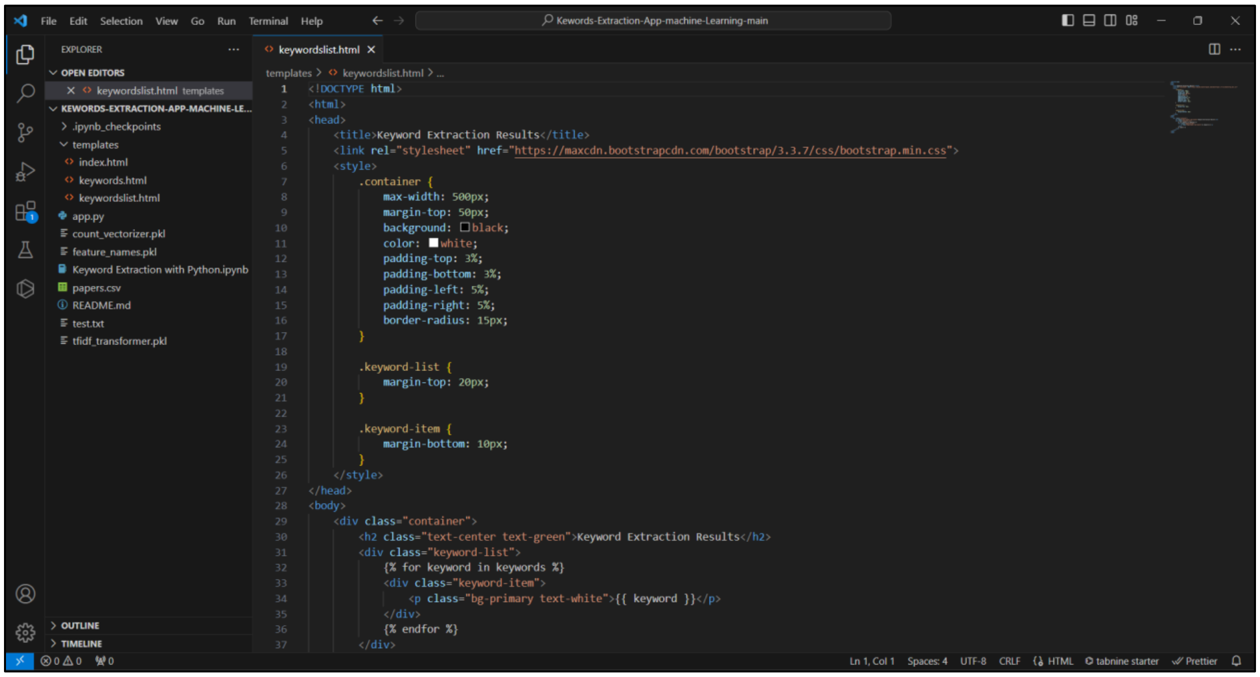
```html
<!DOCTYPE html>
<html>
<head>
    <title>Keyword Extraction</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <style>
        .container {
            max-width: 500px;
            margin-top: 50px;
            background: black;
            color: white;
            padding-top:3%;
            padding-bottom:3%;
            padding-left:5%;
            padding-right:5%;
            border-radius:15px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2 class="text-center text-green">Keyword Extraction</h2>
        <form method="POST" action="/extract_keywords" enctype="multipart/form-data">
            <div class="form-group">
                <label for="document">Upload Document:</label>
                <input type="file" name="file" class="form-control" id="document">
            </div>
            <button type="submit" class="btn btn-primary">Extract Keywords</button>
        </form>
        <hr>
        <form method="POST" action="/search_keywords">
            <div class="form-group">
                <label for="search">Search Keywords:</label>
                <input type="text" name="search" class="form-control" id="search">
            </div>
            <button type="submit" class="btn btn-primary">Search</button>
        </form>
```

Keywords.html



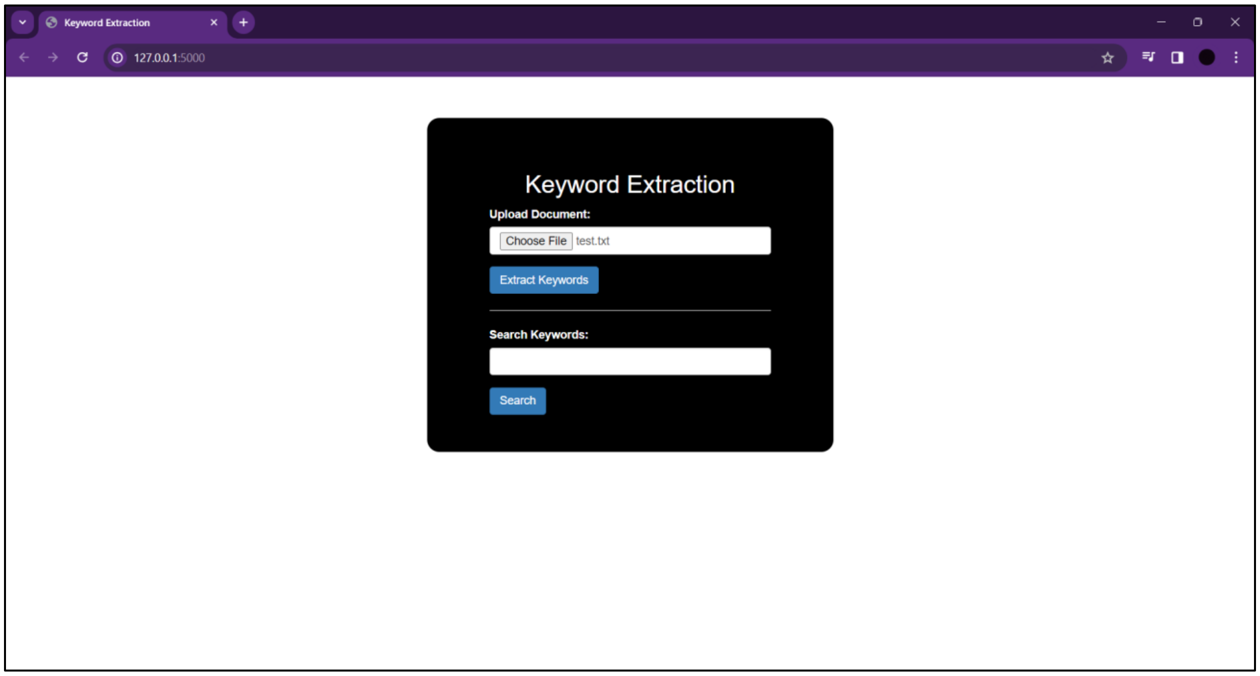
```html
<!DOCTYPE html>
<html>
<head>
    <title>Keyword Extraction Results</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <style>
        .container {
            max-width: 500px;
            margin-top: 50px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Keyword Extraction Results</h2>
        <table class="table table-dark table-hover">
            <thead>
                <tr>
                    <th>Keyword</th>
                    <th>Score</th>
                </tr>
            </thead>
            <tbody>
                {% for keyword, score in keywords.items() %}
                <tr>
                    <td>{{ keyword }}</td>
                    <td>{{ score }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
</body>
</html>
```

Keywordslist.html



```html
<!DOCTYPE html>
<html>
<head>
    <title>Keyword Extraction Results</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <style>
        .container {
            max-width: 500px;
            margin-top: 50px;
            background: black;
            color: white;
            padding-top: 3%;
            padding-bottom: 3%;
            padding-left: 5%;
            padding-right: 5%;
            border-radius: 15px;
        }

        .keyword-list {
            margin-top: 20px;
        }

        .keyword-item {
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2 class="text-center text-green">Keyword Extraction Results</h2>
        <div class="keyword-list">
            {% for keyword in keywords %}
            <div class="keyword-item">
                <p class="bg-primary text-white">{{ keyword }}</p>
            </div>
            {% endfor %}
        </div>
```

Output:



Keyword Extraction

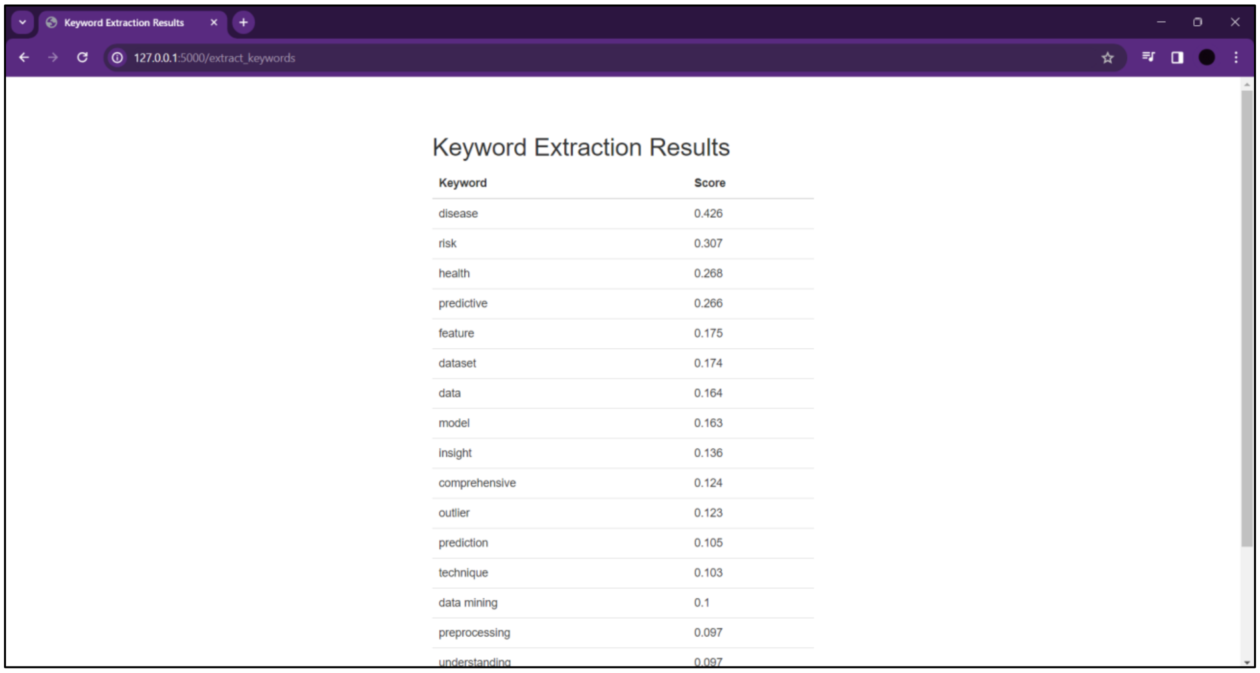
Upload Document:

Choose File test.txt

Extract Keywords

Search Keywords:

Search



Keyword Extraction Results

Keyword	Score
disease	0.426
risk	0.307
health	0.268
predictive	0.266
feature	0.175
dataset	0.174
data	0.164
model	0.163
insight	0.136
comprehensive	0.124
outlier	0.123
prediction	0.105
technique	0.103
data mining	0.1
preprocessing	0.097
understanding	0.097

CHAPTER 6
REFERENCES

1. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

2. Scikit-learn: Machine Learning in Python. (https://scikit-learn.org/stable/)

3. NLTK Documentation: Natural Language Toolkit. (https://www.nltk.org/)

4. Flask Documentation: Web Development with Python. (https://flask.palletsprojects.com/)

5. Pandas Documentation: Python Data Analysis Library. (https://pandas.pydata.org/)