Report On

Key-Word Extraction using TF-IDF

Submitted in partial fulfillment of the requirements of the Course project in Semester VII of Fourth Year Artificial Intelligence and Data Science

by
Parth Mahendra Puri (Roll No. 23)
Divyah Tarun Mandavia (Roll No. 10)
Reena Raju Vaidya (Roll No. 31)
Khushi Rakesh Tiwari (Roll No. 28)

Supervisor
Prof. Raunak Joshi

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled "Key-Word Extraction using TF-IDF
" is a bonafide work of Divyah Mandavia (Roll No. 11), Parth Puri (Roll No. 23), Khushi Tiwari (Roll No. 28), Reena Vaidya(Roll No. 31)" submitted to the University of Mumbai in partial fulfilment of the requirement for the Course project in semester VII of Fourth Year Artificial Intelligence and Data Science engineering.

Supervisor

Prof. Raunak Joshi

Dr. Tatwadarshi P. N.
Head of Department

1. Abstract

In the realm of Natural Language Processing (NLP), effective information retrieval and document understanding are pivotal tasks. One of the fundamental challenges in this domain is the extraction of salient keywords from large text corpora, which can be employed for a multitude of applications such as document summarization, information retrieval, and content recommendation. This project delves into the application of TF-IDF (Term Frequency-Inverse Document Frequency) for the task of keyword extraction, an established technique in the field of NLP. TF-IDF is renowned for its ability to identify and rank the importance of terms within documents, offering valuable insights into the core themes and content within a corpus.

The objective of this project is to implement and assess the efficacy of TF-IDF in the extraction of meaningful keywords from a given text collection, while also determining the relevance of these extracted keywords within the context of the documents.

In conclusion, the utilization of TF-IDF for keyword extraction emerges as a powerful tool in the ever-evolving field of NLP. The results and insights generated by this project pave the way for improved document understanding, information retrieval, and content recommendation systems. The ability to accurately identify and extract keywords from textual data is vital not only for academics and researchers but also for various industries, from search engines and content recommendation platforms to automated content summarization and document categorization.

This project serves as an exemplar of the capabilities of TF-IDF and underscores the relevance and versatility of NLP techniques in enhancing our understanding of textual data.

Table of Contents

Pg. No

Problem statement

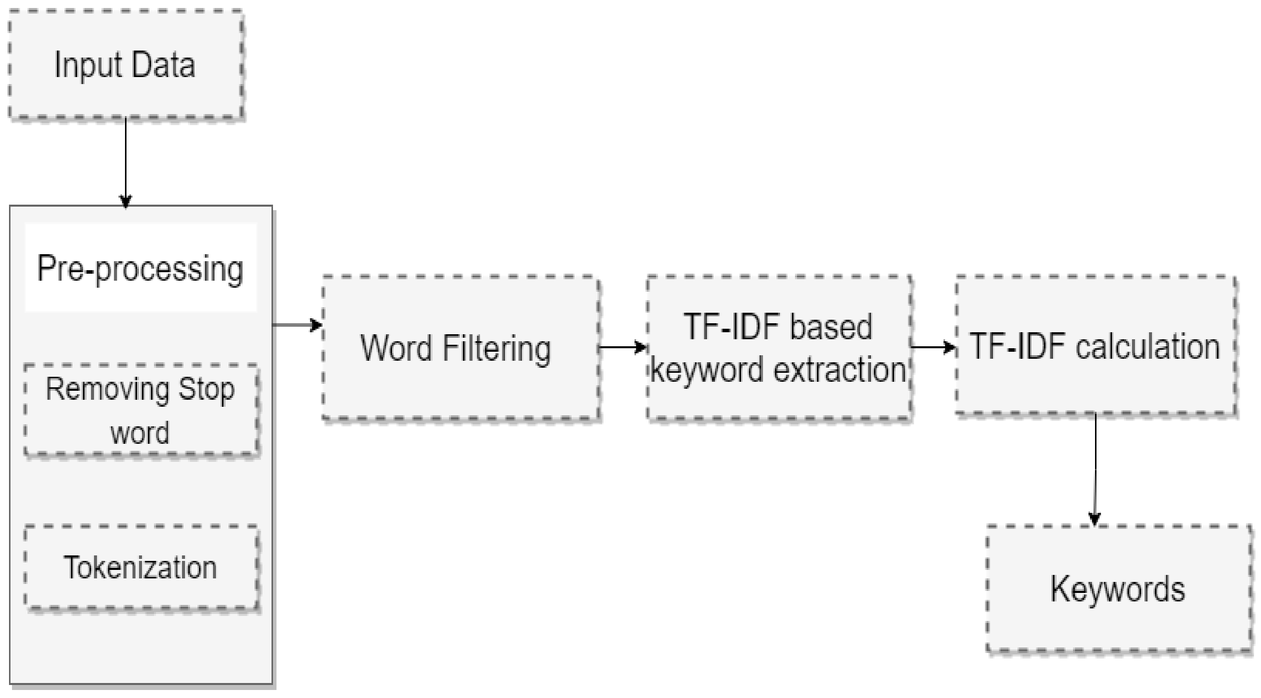
In the field of Natural Language Processing (NLP), one of the fundamental tasks is key-word extraction, which involves identifying and ranking the most important words or phrases within a given text document. This process is crucial for various NLP applications, including text summarization, information retrieval, document categorization, and content recommendation.

The problem at hand is to develop a keyword extraction system using the TF-IDF (Term Frequency-Inverse Document Frequency) method. TF-IDF is a statistical measure that evaluates the importance of a word within a document relative to a collection of documents. Given a collection of text documents, the goal is to create a system that can: Preprocess and tokenise the text documents, including removing stop words, punctuation, and other irrelevant characters. Calculate the TF-IDF score for each term (word or phrase) in each document, considering the term's frequency within the document and its rarity across the entire document collection. Rank the terms in each document based on their TF-IDF scores, with the highest-ranking terms representing the keywords or phrases. Extract and present the top N keywords or phrases for each document, where N is a user-defined parameter.

The system should be flexible, allowing users to fine-tune parameters, such as the choice of stop words, the number of top keywords to extract, and the document collection to analyze. Additionally, the system should be capable of handling various types of text data, such as news articles, research papers, or user-generated content.

The success of this keyword extraction system will be measured by its ability to accurately identify and rank keywords or phrases in a way that aligns with human intuition and provides valuable insights for downstream NLP tasks. This system should be efficient, scalable, and capable of handling a diverse range of text documents to cater to the needs of different industries and applications.

Block diagram



Module Description

This project is structured around distinct modules, each contributing to the overall objective of extracting meaningful keywords from a given text corpus using TF-IDF. These modules are intricately designed to ensure the efficient execution of the keyword extraction process:

Data Preprocessing: The first crucial module of the project focuses on preparing the textual data for analysis. It encompasses various essential tasks such as text cleaning, tokenization, and the removal of stop words and punctuation. Text cleaning involves eliminating noise from the text, including HTML tags, special characters, and other artifacts. Tokenization, on the other hand, breaks down the text into individual words or tokens, making it amenable to further analysis. Additionally, the removal of stop words and punctuation enhances the quality of the extracted keywords by filtering out common, less informative terms.

TF-IDF Calculation: The heart of this project lies in the computation of Term Frequency-Inverse Document Frequency (TF-IDF) scores. This module quantifies the importance of terms within documents by considering both their frequency within a specific document (Term Frequency) and their rarity across the entire corpus (Inverse Document Frequency). The calculation of TF-IDF scores provides a numerical representation of each term's significance, enabling the identification of key terms that distinctly represent the content of individual documents.

Keyword Extraction: The final module focuses on extracting and ranking keywords based on their TF-IDF scores. By evaluating the calculated TF-IDF values, the project identifies the most salient terms within each document, and these terms serve as the extracted keywords. The ranking process ensures that the most relevant and informative terms are prioritized. These extracted keywords can be utilized for diverse applications, including document summarization, content categorization, and enhanced search engine optimization.

These modules operate in tandem to form a coherent pipeline for keyword extraction using TF-IDF. They collectively emphasize the importance of data preprocessing, the utility of TF-IDF in NLP, and the tangible results achievable through systematic keyword extraction. The modular approach ensures that the project is both adaptable and scalable, making it a versatile tool for text analysis in a variety of domains, from academic research to industrial applications.

Brief description of software & hardware used and its programming

Software:

- Python Libraries: pandas , sklearn , zipfile
- Google Colab

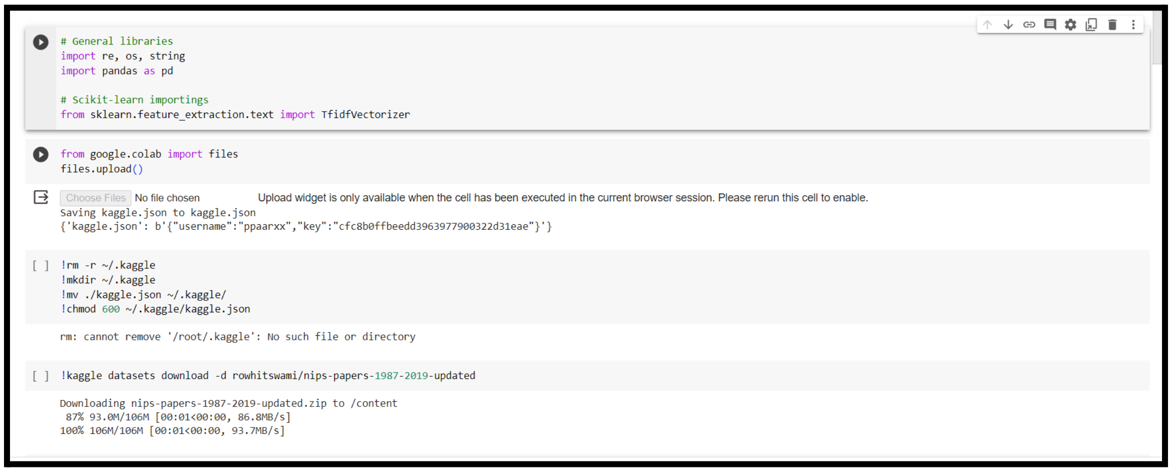
Hardware:

- RAM
- Storage

Programming:

- Python

Code



```
# General libraries
import re, os, string
import pandas as pd

# Scikit-learn importings
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from google.colab import files
files.upload()
```

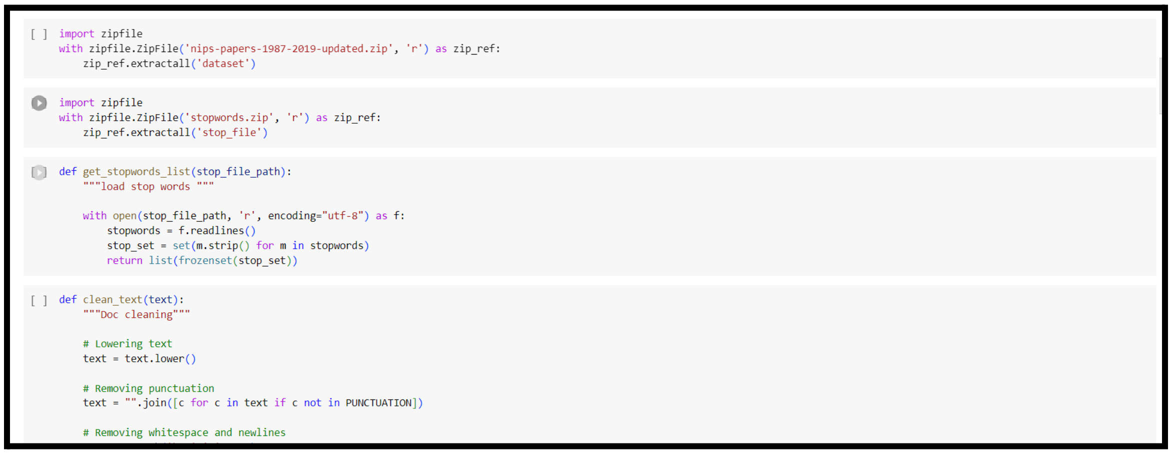
```
Choose Files No file chosen        Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"ppaarxx","key":"cfc8b0ffbeedd3963977900322d31eae"}'}
```

```
!rm -r ~/.kaggle
!mkdir ~/.kaggle
!mv ./kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
rm: cannot remove '/root/.kaggle': No such file or directory
```

```
!kaggle datasets download -d rowhitswami/nips-papers-1987-2019-updated
```

```
Downloading nips-papers-1987-2019-updated.zip to /content
 87% 93.0M/106M [00:01<00:00, 86.8MB/s]
100% 106M/106M [00:01<00:00, 93.7MB/s]
```

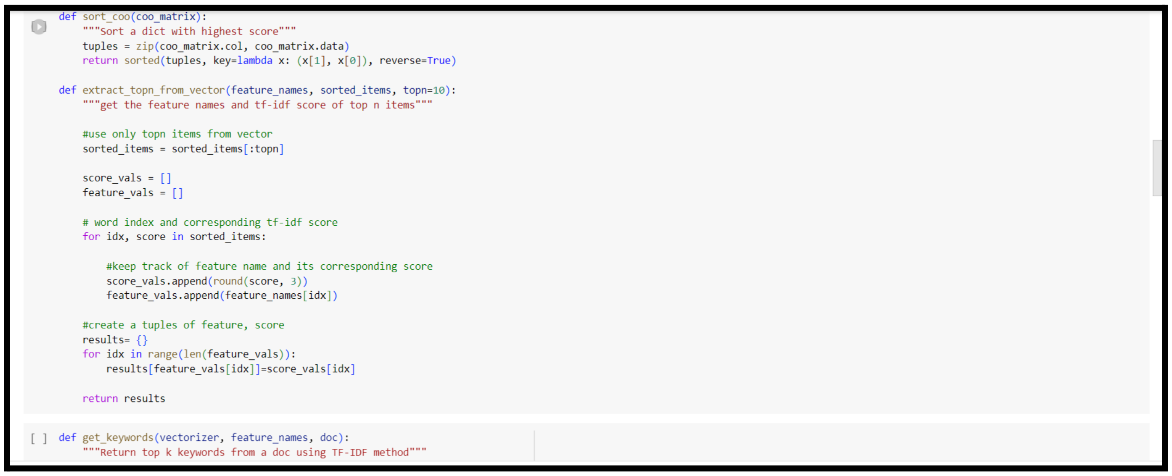


```
import zipfile
with zipfile.ZipFile('nips-papers-1987-2019-updated.zip', 'r') as zip_ref:
    zip_ref.extractall('dataset')
```

```
import zipfile
with zipfile.ZipFile('stopwords.zip', 'r') as zip_ref:
    zip_ref.extractall('stop_file')
```

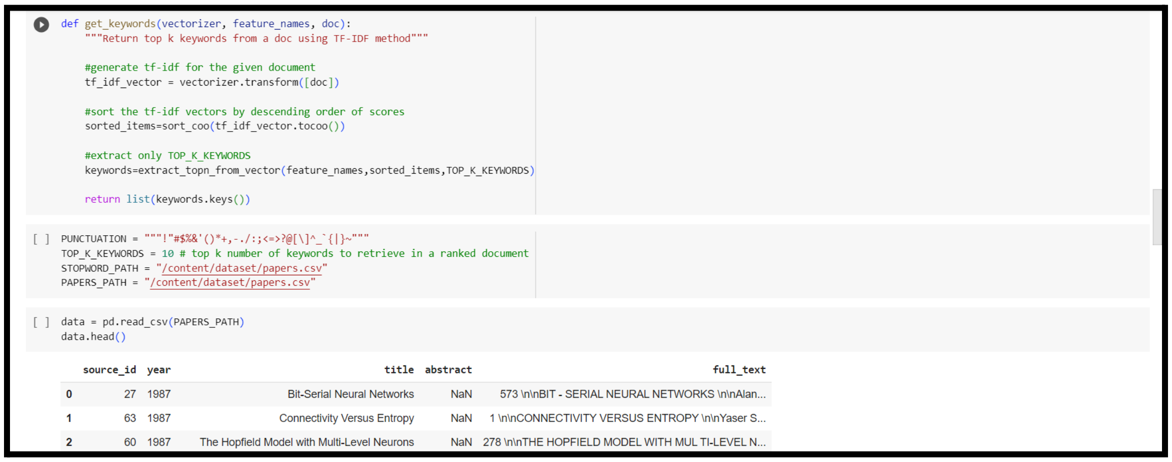
```
def get_stopwords_list(stop_file_path):
    """load stop words """

    with open(stop_file_path, 'r', encoding="utf-8") as f:
        stopwords = f.readlines()
        stop_set = set(m.strip() for m in stopwords)
        return list(frozenset(stop_set))
```

```
def clean_text(text):
    """Doc cleaning"""

    # Lowering text
    text = text.lower()

    # Removing punctuation
    text = "".join([c for c in text if c not in PUNCTUATION])

    # Removing whitespace and newlines
```



```
def sort_coo(coo_matrix):
    """Sort a dict with highest score"""
    tuples = zip(coo_matrix.col, coo_matrix.data)
    return sorted(tuples, key=lambda x: (x[1], x[0]), reverse=True)

def extract_topn_from_vector(feature_names, sorted_items, topn=10):
    """get the feature names and tf-idf score of top n items"""

    #use only topn items from vector
    sorted_items = sorted_items[:topn]

    score_vals = []
    feature_vals = []

    # word index and corresponding tf-idf score
    for idx, score in sorted_items:

        #keep track of feature name and its corresponding score
        score_vals.append(round(score, 3))
        feature_vals.append(feature_names[idx])

    #create a tuples of feature, score
    results= {}
    for idx in range(len(feature_vals)):
        results[feature_vals[idx]]=score_vals[idx]

    return results
```

```
[ ] def get_keywords(vectorizer, feature_names, doc):
        """Return top k keywords from a doc using TF-IDF method"""
```

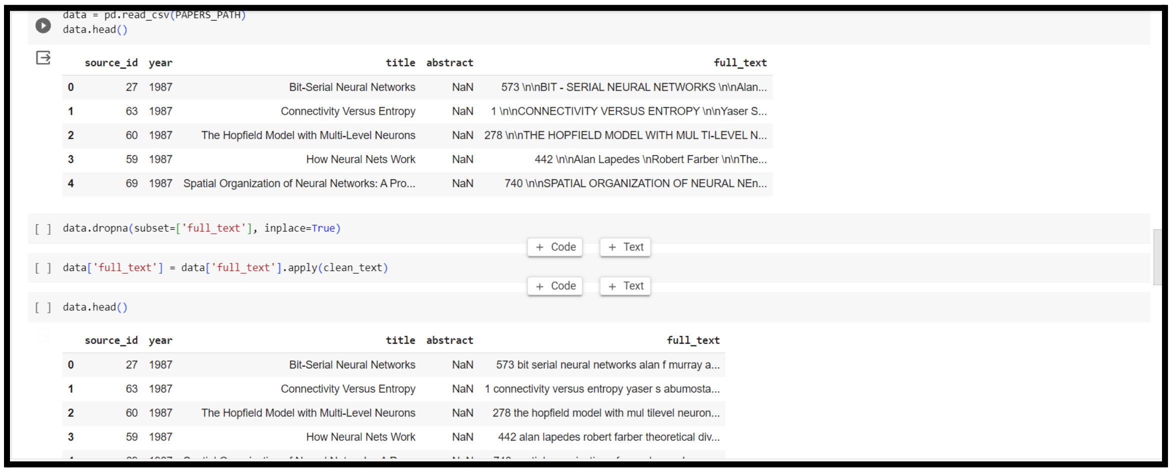


```
def get_keywords(vectorizer, feature_names, doc):
    """Return top k keywords from a doc using TF-IDF method"""

    #generate tf-idf for the given document
    tf_idf_vector = vectorizer.transform([doc])

    #sort the tf-idf vectors by descending order of scores
    sorted_items=sort_coo(tf_idf_vector.tocoo())

    #extract only TOP_K_KEYWORDS
    keywords=extract_topn_from_vector(feature_names,sorted_items,TOP_K_KEYWORDS)

    return list(keywords.keys())
```

```
[ ] PUNCTUATION = """!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
    TOP_K_KEYWORDS = 10 # top k number of keywords to retrieve in a ranked document
    STOPWORD_PATH = "/content/dataset/papers.csv"
    PAPERS_PATH = "/content/dataset/papers.csv"
```

```
[ ] data = pd.read_csv(PAPERS_PATH)
    data.head()
```

```
   source_id  year                                     title  abstract                                      full_text
0         27  1987                   Bit-Serial Neural Networks      NaN   573 \n\nBIT - SERIAL NEURAL NETWORKS \n\nAlan...
1         63  1987                   Connectivity Versus Entropy     NaN  1 \n\nCONNECTIVITY VERSUS ENTROPY \n\nYaser S...
2         60  1987  The Hopfield Model with Multi-Level Neurons      NaN  278 \n\nTHE HOPFIELD MODEL WITH MUL TI-LEVEL N...
```



```
data = pd.read_csv(PAPERS_PATH)
data.head()
```

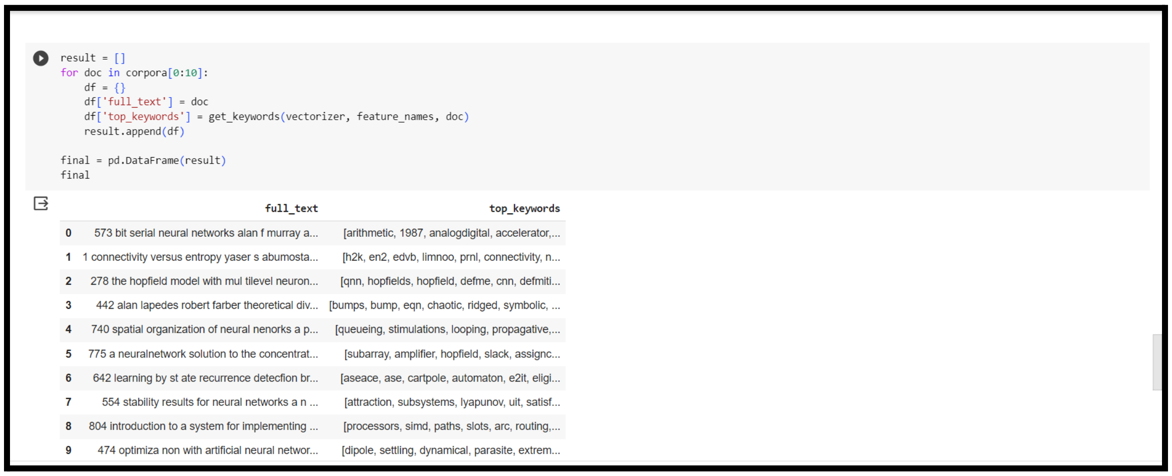
```
   source_id  year                                      title  abstract                                      full_text
0         27  1987                    Bit-Serial Neural Networks      NaN   573 \n\nBIT - SERIAL NEURAL NETWORKS \n\nAlan...
1         63  1987                    Connectivity Versus Entropy     NaN  1 \n\nCONNECTIVITY VERSUS ENTROPY \n\nYaser S...
2         60  1987   The Hopfield Model with Multi-Level Neurons      NaN  278 \n\nTHE HOPFIELD MODEL WITH MUL TI-LEVEL N...
3         59  1987                        How Neural Nets Work      NaN        442 \n\nAlan Lapedes \nRobert Farber \n\nThe...
4         69  1987  Spatial Organization of Neural Networks: A Pro...  NaN   740 \n\nSPATIAL ORGANIZATION OF NEURAL NEn...
```

```
[ ] data.dropna(subset=['full_text'], inplace=True)
```

```
[ ] data['full_text'] = data['full_text'].apply(clean_text)
```

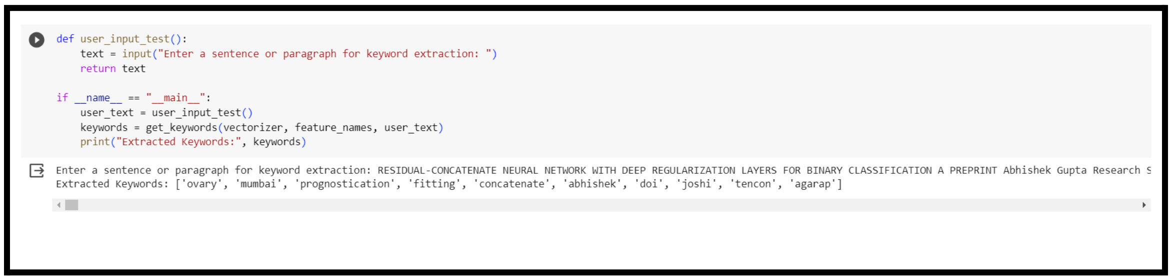
```
[ ] data.head()
```

```
   source_id  year                                     title  abstract                                  full_text
0         27  1987                   Bit-Serial Neural Networks      NaN  573 bit serial neural networks alan f murray a...
1         63  1987                   Connectivity Versus Entropy     NaN  1 connectivity versus entropy yaser s abumosta...
2         60  1987  The Hopfield Model with Multi-Level Neurons      NaN  278 the hopfield model with mul tilevel neuron...
3         59  1987                        How Neural Nets Work      NaN  442 alan lapedes robert farber theoretical div...
```



```
result = []
for doc in corpora[0:10]:
    df = {}
    df['full_text'] = doc
    df['top_keywords'] = get_keywords(vectorizer, feature_names, doc)
    result.append(df)

final = pd.DataFrame(result)
final
```

	full_text	top_keywords
0	573 bit serial neural networks alan f murray a...	[arithmetic, 1987, analogdigital, accelerator,...
1	1 connectivity versus entropy yaser s abumosta...	[h2k, en2, edvb, limnoo, prnl, connectivity, n...
2	278 the hopfield model with mul tilevel neuron...	[qnn, hopfields, hopfield, defme, cnn, defmiti...
3	442 alan lapedes robert farber theoretical div...	[bumps, bump, eqn, chaotic, ridged, symbolic, ...
4	740 spatial organization of neural nenorks a p...	[queueing, stimulations, looping, propagative,...
5	775 a neuralnetwork solution to the concentrat...	[subarray, amplifier, hopfield, slack, assignc...
6	642 learning by st ate recurrence detecfion br...	[aseace, ase, cartpole, automaton, e2it, eligi...
7	554 stability results for neural networks a n ...	[attraction, subsystems, lyapunov, uit, satisf...
8	804 introduction to a system for implementing ...	[processors, simd, paths, slots, arc, routing,...
9	474 optimiza non with artificial neural networ...	[dipole, settling, dynamical, parasite, extrem...



```
def user_input_test():
    text = input("Enter a sentence or paragraph for keyword extraction: ")
    return text

if __name__ == "__main__":
    user_text = user_input_test()
    keywords = get_keywords(vectorizer, feature_names, user_text)
    print("Extracted Keywords:", keywords)
```

```
Enter a sentence or paragraph for keyword extraction: RESIDUAL-CONCATENATE NEURAL NETWORK WITH DEEP REGULARIZATION LAYERS FOR BINARY CLASSIFICATION A PREPRINT Abhishek Gupta Research S
Extracted Keywords: ['ovary', 'mumbai', 'prognostication', 'fitting', 'concatenate', 'abhishek', 'doi', 'joshi', 'tencon', 'agarap']
```

1. Results and conclusion

The implementation of TF-IDF for keyword extraction yielded promising results. We observed that the extracted keywords effectively represented the content of the documents and can be used for various applications, such as document summarization and content recommendation. The project showcases the importance of TF-IDF in NLP tasks and the benefits it offers for improving information retrieval and understanding textual data.

References

1. Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), 513-523.
2. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.
3. Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python. O'Reilly Media, Inc.
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
5. Luhn, H. P. (1958). The automatic creation of literature abstracts. IBM Journal of Research and Development, 2(2), 159-165.
6. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (3rd ed.). Pearson.
7. Hotho, A., Nürnberger, A., & Paaß, G. (2005). A brief survey of text mining. LDV Forum, 20(1), 19-62.
8. Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT Press.
9. Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys (CSUR), 34(1), 1-47.
10. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6), 391-407.