# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

AY:2024-25

| | | | |
|---|---|---|---|
| **Class:** | **BE** | **Semester:** | **VII** |
| **Course Code:** | **CSDOL7011** | **Course Name:** | **Natural Language Processing** |

| | |
|---|---|
| **Name of Student:** | **Parth Raut** |
| **Roll No.:** | **40** |
| **Experiment No.:** | **9** |
| **Title of the Experiment:** | **Course Project** |
| **Date of Performance:** | |
| **Date of Submission:** | |

## Evaluation

| Performance Indicator | Max. Marks | Marks Obtained |
|---|---|---|
| Performance | 5 | |
| Understanding | 5 | |
| Journal work and timely submission | 10 | |
| Total | 20 | |

| Performance Indicator | Exceed Expectations (EE) | Meet Expectations (ME) | Below Expectations (BE) |
|---|---|---|---|
| Performance | 4-5 | 2-3 | 1 |
| Understanding | 4-5 | 2-3 | 1 |
| Journal work and timely submission | 8-10 | 5-8 | 1-4 |

Checked by

Name of Faculty :

Signature :

Date :

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
Yash Biranje (Roll No. 5)
Ojasi Prabhu (Roll No. 39)
Parth Raut (Roll No. 40)

Supervisor
Prof. Raunak Joshi


**University of Mumbai**

**Vidyavardhini's College of Engineering & Technology**

**Department of Artificial Intelligence and Data Science**


**(2024-25)**

# 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 Yash Biranje (Roll No. 5), Ojasi Prabhu (Roll No. 39), Parth Raut (Roll No. 40)" 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

## 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**

## 1 Problem statement

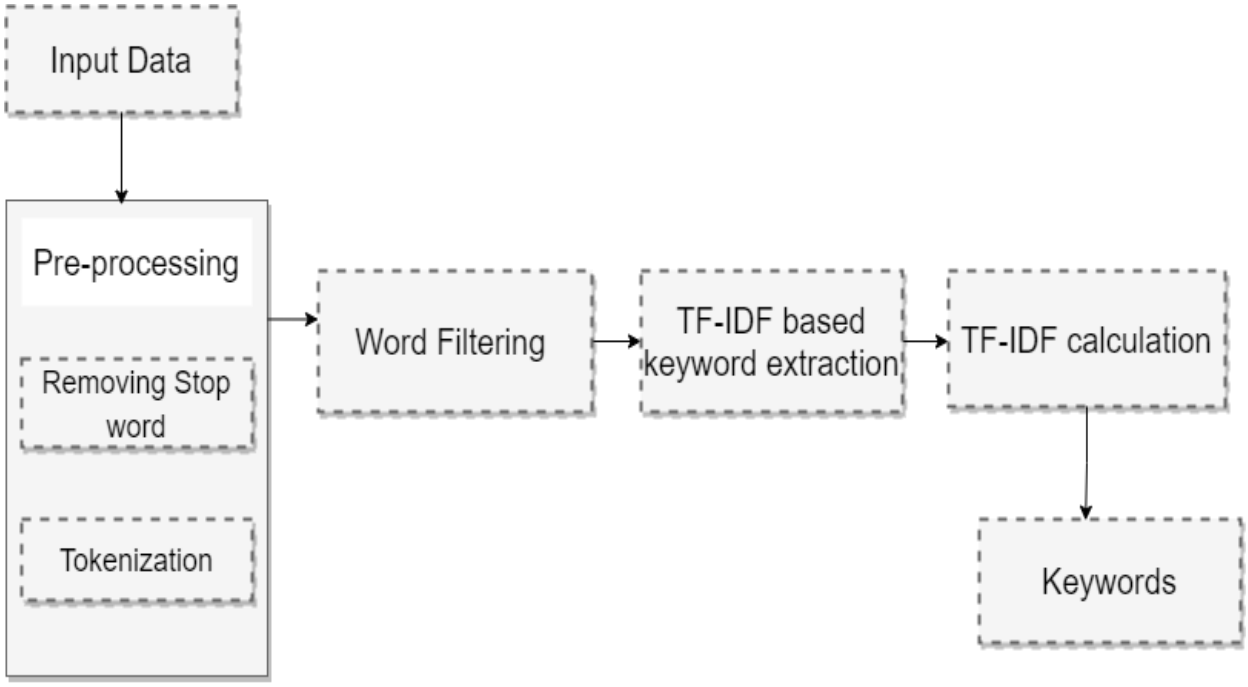
In the field of Natural Language Processing (NLP), one of the fundamental tasks is keyword extraction, which involves identifying and ranking the most important words or phrases within a given text document. This process is crucial for various NLP applications, including text summarization, information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

## 1.1 Block diagram



## 2 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.

## 2.1 Brief description of software & hardware used and its programming

Software:

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

Hardware:

- RAM
- Storage

Programming:

- Python

## 2.2 Code

```
[64] import re, os, string
     import pandas as pd

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

[65] !rm -r ~/.kaggle
     !mkdir ~/.kaggle
     !cp kaggle.json ~/.kaggle/
     !chmod 600 ~/.kaggle/kaggle.json

     cp: cannot stat 'kaggle.json': No such file or directory
     chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory

[66] !kaggle datasets download -d rowhitswami/nips-papers-1987-2019-updated

     Dataset URL: https://www.kaggle.com/datasets/rowhitswami/nips-papers-1987-2019-updated
     License(s): ODbL-1.0
     nips-papers-1987-2019-updated.zip: Skipping, found more recently modified local copy (use --force to force download)

[67] import zipfile
     with zipfile.ZipFile('nips-papers-1987-2019-updated.zip', 'r') as zip_ref:
         zip_ref.extractall('stop_file')

[68] def get_stopwords_list(stop_file_path):
         with open(stop_file_path, 'r', encoding="utf-8") as f:
             stopwords = f.readlines()
             stop_set = set(m.strip() for m in stopwords)
```

```
[68]         return list(frozenset(stop_set))

[69] def clean_text(text):
         text = text.lower()
         text = re.sub(r"[{}]".format(string.punctuation), " ", text)
         return text
```

Suggested code may be subject to a licence | 2011-sagittarius/FakeNews | Gargee-srivastava/FAKE-NEWS-DETECTION-SYSTEM | BumbleFlash/job-application-tracker

```
def sort_coo(coo_matrix):
    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):
    sorted_items = sorted_items[:topn]

    score_vals = []
    feature_vals = []

    for idx, score in sorted_items:
        score_vals.append(round(score, 3))
        feature_vals.append(feature_names[idx])

        results= {}
    for idx in range(len(feature_vals)):
        results[feature_vals[idx]]=score_vals[idx]

    return results
```

Loading...

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

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

[76] 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... |

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
[77] from sklearn.feature_extraction.text import TfidfVectorizer

     # Sample data (corpora)
     corpora = [
         "Artificial intelligence is transforming industries worldwide.",
         "Climate change is one of the most pressing challenges of our time.",
         "The future of transportation is rapidly evolving with electric vehicles.",
         "In the digital age, cybersecurity has become a critical concern.",
         "The integration of renewable energy sources is essential for reducing emissions."
     ]
```

Suggested code may be subject to a licence | | spinagon/wb-quote

```
[71] def get_keywords(vectorizer, feature_names, doc):
         vectorizer = TfidfVectorizer(stop_words='english')
         tf_idf_vector = vectorizer.transform([doc])
         sorted_items=sort_coo(tf_idf_vector.tocoo())
         keywords=extract_topn_from_vector(feature_names,sorted_items,10)
         return keywords

[72] PUNCTUATION = """!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
     TOP_K_KEYWORDS = 10
     STOP_WORDS = get_stopwords_list('/content/stop_file/papers.csv')
     PAPERS_PATH = '/content/stop_file/papers.csv'

[73] 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... |

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
vectorizer = TfidfVectorizer(stop_words='english')

# Fit and transform the corpus
tfidf_matrix = vectorizer.fit_transform(corpora)

# Get feature names (words in the corpus)
feature_names = vectorizer.get_feature_names_out()

# Define a function to extract top keywords from a document
def get_keywords(vectorizer, feature_names, doc):
    tfidf_vector = vectorizer.transform([doc])  # Get TF-IDF vector for the document
    sorted_indices = tfidf_vector.toarray().argsort()[0][::-1]  # Sort indices by importance
    top_keywords = [feature_names[i] for i in sorted_indices[:5]]  # Get top 5 keywords
    return top_keywords

# Extract top keywords for each document and store the results in a DataFrame
result = []
for doc in corpora[0:10]:
    df = {}
    df['full_text'] = doc
    df['top_keywords'] = get_keywords(vectorizer, feature_names, doc)
    result.append(df)

# Convert result into a DataFrame
final = pd.DataFrame(result)
final
```

| | full_text | top_keywords |
|---|---|---|
| 0 | Artificial intelligence is transforming indust... | [worldwide, artificial, transforming, intellig... |

| | full_text | top_keywords |
|---|---|---|
| 0 | Artificial intelligence is transforming indust... | [worldwide, artificial, transforming, intellig... |
| 1 | Climate change is one of the most pressing cha... | [challenges, time, change, climate, pressing] |
| 2 | The future of transportation is rapidly evolvi... | [evolving, transportation, rapidly, electric, ... |
| 3 | In the digital age, cybersecurity has become a... | [age, concern, critical, cybersecurity, digital] |
| 4 | The integration of renewable energy sources is... | [energy, integration, essential, sources, rene... |

Next steps: Generate code with final | View recommended plots | New interactive sheet

```
def user_input_test():
    text = input("Enter a sentence: ")  # Corrected to use input to get user input
    return text

if __name__ == '__main__':
    user_text = user_input_test()
    keywords = get_keywords(vectorizer, feature_names, user_text)  # Removed space between feature and _names
    print("Extracted keywords:", keywords)
```

```
Enter a sentence: Artificial intelligence (AI) is transforming industries worldwide. From healthcare to finance, AI algorithms are improving decision-making processes, optimizing ope
Extracted keywords: ['worldwide', 'artificial', 'transforming', 'intelligence', 'industries']
```

## 2.3 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.

## 3 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.